Západočeská univerzita v Plzni
Fakulta aplikovaných věd
Katedra informatiky a výpočetní techniky

Diplomová práce

Využití PowerShellu pro správu Windows sítě

Plzeň, 2011 Václav Dušek

Zde bude vloženo originální zadání práce.

Prohlášení

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů.

V Plzni dne 23. června 2011 ….……

Václav Dušek

Abstract

Usage of PowerShell for Windows network management

PowerShell is extensible shell with scripting language for Windows. The objective of diploma thesis is to investigate its options in terms of Windows network management and Active directory domain. Next aim is suggest appropriate activities for network management that will be used and create an application that these activities will be controlled. For this application was chosen web interface, that contains five modules for network management. Application is also extensible and adding next modules or modify current modules is possible. The functionality of the application will be verified in the virtual environment.

Obsah

Abstract 4
1 Úvod 7
2 PowerShell 8
2.1 Předchůdci 8
2.2 Ovládání PowerShellu 10
2.2.1 Cmdlety 10
2.2.2 Funkce 12
2.2.3 Spustitelné programy a skripty 14
2.3 Roura 14
2.3.1 Textová roura 14
2.3.2 Objektová roura 15
2.4 Uživatelské profily 17
2.5 Aliasy 17
2.6 Práce s daty 18
2.6.1 Proměnné 18
2.6.2 Datové typy 19
2.7 Prostředí 19
2.7.1 PowerShell ISE 19
2.7.2 PowerGui 20
2.8 Rozšíření a doplňky 20
2.8.1 PowerShell Comunity Extensions 21
2.8.2 ActiveRoles Management Shell for Active Directory 21
2.8.3 NetCmdlets 21
2.8.4 PowerGadgets 22
2.8.5 Windows System Modules 22
3 Přístup k rozhraním 23
3.1 Active Directory 23
3.1.1 Rozhraní ADSI 23
3.1.2 Vnější struktura Active Directory 24
3.1.2 Vnitřní struktura Active Directory 24
3.1.3 Přístup z PowerShellu 25
3.2 WMI (Windows Management Instrumentation) 25
3.2.1 Získávání informací 26
3.2.2 Volání akcí 27
4 Vybrané části PowerShellu 28
4.1 Logy 28
4.2 Služby 28
4.3 Procesy 29
5 Analýza vhodných činnosti pro správu 30
5.1 Zvolené činnosti 30
5.2 Využití webových služeb 30
5.3 Získaná data 32
6 Návrh systému 34
6.1 Spouštění z konzole 34

6.2 Webová aplikace ... 35
6.2.1 Generování skriptů pro vytváření uživatelských kont ... 36
6.2.2 Správa Active Directory ... 39
6.2.3 MSDN ... 40
6.2.4 WMI ... 42
6.2.5 Služby a procesy ... 42
7 Realizace systému ... 44
7.1 Webové služby ... 44
7.2 Webové rozhraní ... 45
7.3 Rozšiřující funkce ... 47
7.4 PowerShell skripty ... 49
8 Testovací scénáře ... 51
9 Závěr ... 54
LITERATURA A ZDROJE ... 55
PŘÍLOHY ... 58
A – Instalační příručka ... 58
B – Uživatelská příručka ... 59
C – Obsah přiloženého CD ... 63

1 Úvod

V počátcích operačních systémů spočívala správa výpočetních systémů zadáváním příkazů do textové konzole. Takto nějak vznikal shell a obecně skriptování. V dalším postupném vývoji operačních systémů byla správa výpočetních systémů prostřednictvím příkazové řádky používána u operačních systémů Unix, zatímco operační systémy Mirosoft Windows nabízely grafické uživatelské rozhraní. V roce 2003 však i společnost Microsoft začíná vyvíjet vlastní shell. V roce 2006 uvolňuje první verzi PowerShellu pod kódovým názvem Monad. Jedná se o rozšiřitelný shell se skriptovacím jazykem s dobrými možnostmi pro administrátorskou správu výpočetních systémů. Nově vzniklá technologie PowerShell je postavena na základní koncepci Microsoft Windows, kterou je rozhraní .NET Framework. Z této skutečnosti vyplývá i fakt, že PowerShell je objektově orientovaným skriptovacím jazykem. Cílem Microsoftu je vytvořit moderní ovládací prostředek pro správu typicky serverových aplikací, nejen operačních systémů.

Cílem této práce je vytvořit skripty pro správu Windows serveru a navrhnout jako webovou aplikaci uživatelsky přívětivé prostředí, které snadno umožní správu navržených oblastí systému bez nutnosti znalosti PowerShellu. Skripty budou zaměřeny na různé oblasti. Webové uživatelské rozraní umožní spravovat objekty uložené v Active Directory, zjišťovat stav serverových součástí, správu služeb a procesů na serveru a umožní také zjištění konkrétních uživatelských kont již nestudujících studentů na katedře informatiky a výpočetní techniky (dále KIV). Přístup do uživatelského prostředí bude chráněn heslem, neboť umožní správu důležitých komponent v celé počítačové síti.

2 PowerShell

Technologie PowerShell je rozšiřitelný shell se skriptovacím jazykem postaveným na platformě .NET Framework. Díky tomu se výrazně liší od svých předchůdců, protože má k dispozici objektovou rouru místo standardní textové. Navíc dokáže využít veškeré přednosti .NET Frameworku. Díky této provázanosti má PowerShell k dispozici velké množství funkcí pro správu prostřednictvím takzvaných cmdletů. Jedná se o třídy .NET implementující určitou operaci. Akceptováno je též spouštění všech programů stejným způsobem jako v příkazové řádce Windows. PowerShell umožňuje též plný přístup ke COM (Component Object Model) a WMI (Windows Management Instrumentation). Všechny tyto prvky lze mezi sebou různě kombinovat. Podobně jako u shellů v Unixu nabízí PowerShell systém nápovědy pro jednotlivé příkazy. PowerShell také poskytuje mechanismus, s nímž může být PowerShell vložen do jiných aplikací. Tyto aplikace pak využívají PowerShell funkce k provedení určité operace, včetně aplikací využívající grafické rozhraní. Tato schopnost byla využita v Microsoft Exchange Server 2007.

První verze PowerShellu byla vydána v roce 2006 a do starších systémů od verze Windows XP a v serverovém případě od verze Windows Server 2003 jej bylo možné přidat pomocí Service Packů. Současná verze 2.0 byla uvolněna v létě roku 2009. Ta již je pevnou součástí operačních systémů Windows 7 a Windows Server 2008 R2.

2.1 Předchůdci

V minulosti shelly ve Windows neobsahovaly žádný propracovaný a silný skriptovací jazyk. Pokročilí uživatelé a správci se tak museli spoléhat na řadu dalších cest, jak tuto nevýhodu obejít. V následujících bodech je přehled technologií, které předcházeli nástupu PowerShellu.

Příkazová prostředí

První příkazové rozhraní bylo k dispozici u operačního systému MS-DOS. Jednalo se o jediné rozhraní pro ovládání prvního operačního systému od Microsoftu. Neobsahoval skriptovací jazyk, nabízel základní příkazy.

Novější a dodnes používaná varianta příkazového řádku *cmd.exe* vychází ze svého předchůdce a nastupuje v době vývoje 32bitových operačních systémů. V dnešní době je výchozí součástí systémů Windows. Ani zde však není obsažen skriptovací jazyk a potenciál zůstává značně omezený. Už se zde objevuje sada konzolových aplikací, která vylepšuje možnosti správy služeb, procesů, registrů apod.

Platforma Windows Script Host

Jedná se o konzistentní a všestranné skriptovací prostředí. Společnost Microsoft poskytuje dva výchozí jazyky Microsoft Visual Basic Script a Java Script, které přinesly celou řadu do té doby chybějících konstrukcí. Do platformy WSH existuje možnost přidání dalších jazyků. Mezi nejčastější patří Perl a Python, ale také třeba PHP. Platforma WSH je instalována ve výchozím nastavení systému Windows 98 a novějších verzích systému Windows. Též je instalována s verzí Internet Explorer 5 a vyšší. Od operačního systému Windows 2000 WSH umožňuje použití přihlašovacích skriptů uživatelů.

Skriptovací platformy jiných dodavatelů

V této oblasti stojí za zmínění řešení *KiXtart*. Jedná se o velmi dobře použitelné skriptovací rozhraní pro administraci. Jeho vývoj začal v roce 1991 s cílem poskytnout přihlašovací skripty pro prostředí Microsoft LAN Manager. Má řadu zajímavých vestavěných funkcí a poskytuje přístup k ADSI, ADO, WMI a dalším. Umožňuje spouštět programy, připojit síťové disky, číst nebo upravovat registry a celou řadu dalších možností. Grafickým plug-inem pro KiXtart je KiXforms.

V kapitole 2.1 bylo čerpáno z [1].

2.2 Ovládání PowerShellu

Příkazové prostředí PowerShellu je v operačním systému Windows zabudováno jako jeden spustitelný soubor, nazvaný *powershell.exe*, a lze jej spustit celkem třemi standardními způsoby. Buďto přes dialogové okno Spustit (Run), výběrem v nabídce Start, nebo spuštěním přes standardní konzolové okno *cmd.exe*. Jak již bylo zmíněno, PowerShell nepracuje s grafickými prvky a jako své komunikační rozhraní využívá klasické textové konzole.

Při své činnosti PowerShell zpracovává vstupní řetězce, na základě kterých provede příslušné vyhodnocení. Buď tedy zadaný řetězec vyhodnotí jako srozumitelný a provede konkrétní operaci nebo vypíše podrobné chybové hlášení. Při zadávání vstupního řetězce PowerShell očekává jednu z možností, které jsou popsány v následujících podkapitolách.

2.2.1 Cmdlety

Jedná se o specializovaný příkaz, zcela typický pro PowerShell. Může být napsán v libovolném jazyku, podporovaném platformou .NET. Příkaz Cmdletu je složen ze slovesa a podstatného jména, což umožňuje výstižné pojmenování příkazů vyvolávajících konkrétní operace. Cmdlety běží v procesu samotného PowerShellu a není tak spouštěn žádný další proces. Jejich výhodou je jednotnost použití cmdletů. Přestože je každý určen k jiné činnost, způsob použití je ve všech případech stejný. Oproti svým předchůdcům PowerShell umožňuje sestavovat nové cmdlety, čím si lze své prostředí pro správu obohatit a tím i zdokonalit. Jako výstup po provedení cmdlet vrací objekt nebo kolekci.

Popis syntaxe

```
<příkaz> parametr(určený pozicí) –přepínač
```

Cmdlet umí pracovat s větším množstvím přepínačů nebo také úplně bez nich. Není podmínkou, že přepínač musí obsahovat vstupní hodnotu. Jsou totiž přepínače, které žádné parametry nepotřebují. I většiny přepínačů však hodnota parametru nesmí chybět. Některé parametry mohou být užity i bez přepínače, musí však striktně dodržet

pozici, na které mají být napsány. Pokud přepínač obsahuje více než jeden parametr, musejí být mezi sebou odděleny čárkou.

ukázky

```
Get-Process -Name svc*
```

vypíše všechny běžící procesy začínající řetězcem ‚svc‘

alternativní zápis se stejným výstupem:

```
Get-Process svc*
```

Pořadí přepínačů se zjistí vyvoláním nápovědy příkazu:

`Get-Help Get-Process –Full` *(podobný princip jako i jiných shellů)*

Výsledek nápovědy pro příkaz `Get-Process`:

```
-Name <string[]>
        Specifies one or more processes by process name. You can
type multiple process names (separated by commas) or use
wildcard ch
        aracters. The parameter name ("Name") is optional.
        Required?                    false
        Position?                    1
        Default value
        Accept pipeline input?       true (ByPropertyName)
        Accept wildcard characters?  true
```

Pozn.: řádek určující pozici přepínače je vyznačen tučně

```
Get-Process svc*, fir*
```

vypíše všechny běžící procesy začínající řetězci ‚svc‘ nebo ‚fir‘ , ukázka použití oddělení parametrů čárkou

Tabulka č. 1 ukazuje přehled základních cmdletů a jejich alternativy v jiných shellech. První sloupec uvádí zápis cmdletu v PowerShellu, druhý sloupec jeho alias, třetí sloupec alternativu pro prostředí *cmd.exe* v systému Windows a čtvrtý sloupec zápis v systému na bázi Unix.

PS cmdlet	*PS alias*	*cmd.exe*	*Unix, BSD, Linux*
Get-ChildItem	gci, dir, ls	dir	ls
Get-Content	gc, type, cat	type	cat
Get-Command	gcm	help	which
Get-Help	help, man	help	man
Clear-Host	cls, clear	cls	clear
Copy-Item	cpi, copy, cp	copy	cp
Get-Location	gl, pwd	cd	pwd
Set-Location	sl, cd, chdir	cd	chdir
Write-Output	echo, write	echo	echo
Get-Process	gps, ps	tlist, tasklist	ps
Select-String	n/a	find, findstr	grep

Tabulka č. 2.1: *Přehled nejpoužívanějších cmdletů*

2.2.2 Funkce

Funkce je ucelený blok proveditelného kódu, který se může pod jejím názvem opakovaně spouštět. Princip je zde zcela obdobný jako u jiných jazyků, funkce tedy může mít vstupní parametry a též může vracet návratovou hodnotu.

Ukázka jednoduché funkce pro zjištění volného místa na disku C:, včetně jejího volání a výstupu:

```
function Get-FreeDiskSpace
{(gwmi win32_logicaldisk -filter 'DeviceID="C:"').freespace/1GB}

Get-FreeDiskSpace

10,6497840881348
```

Při definici parametrů funkce je určitě zajímavou možností určit jejich typ. Při deklaraci funkce: `function test {param ([int]$text)}` je zřejmé, že jí půjde volat pouze s celočíselným parametrem.

S verzí 2.0 přichází možnost takzvaných pokročilých funkcí. Ty umožňují několik nových možností jako lepší kontrola vstupních parametrů funkcí nebo

poskytnutí nápovědy pro danou funkci podobně jako u cmdletů. U parametru se tedy naskytují další možnosti, jako nastavit parametr jako povinný (mandatory), nastavit jeho pozici (position) nebo pro parametr vyvolat nápovědu, je-li zadán špatně.

Přehled atributů používaných pro testování hodnot parametrů:

AllowNull	umožňuje, aby parametr obsahoval hodnotu null, platí i pokud je parametr nastaven jako mandatory (povinný)
AllowEmptyString	parametr může být prázdný řetězec
AllowEmptyCollection	stejné jako předchozí dva, ale pro prázdnou kolekci
ValidateCount	určuje minimální a maximální počet argumentů
ValidateLength	určuje délku parametru
ValidatePattern	specifikuje regulární výraz, kterým můžeme určit vzor parametru
ValidateRange	minimální a maximální hodnota parametru
ValidateScript	může obsahovat skript, který validuje parametr, pokud skript vrátí hodnotu false, je zahlášena chyba validace
ValidateSet	parametr může obsahovat pouze uvedené hodnoty
ValidateNotNull	hodnota parametru nemůže být null
ValidateNotNullOrEmpty	hodnota nemůže být null nebo prázdný řetězec

Jednoduchá ukázka, funkce vypíše jméno (povinný textový parametr na 0. pozici, akceptováno je pouze jméno Pavel) a věk (povinný celočíselný parametr na 1. pozici s akceptovanou hodnotou 33)

```
function Test-Parameter
{
   param (
   [Parameter(Mandatory=$true,Position=0)]
   [string][ValidateSet("Pavel")] $jmeno,
   [Parameter(Mandatory=$true,Position=1)] [int]
   [ValidateRange(33,33)] $vek
   )
   Write-Output "Jmenuje se: $jmeno a je mu: $vek let."
}
```

2.2.3 Spustitelné programy a skripty

PowerShell stejně jako příkazové prostředí systému Windows *cmd.exe* umí spustit program, který je k dispozici, ať už se jedná o konzolovou aplikaci (např. síťová konfigurace – ipconfig) nebo jiný externí program.

Další možností spuštění externího souboru je PowerShell skript. Tyto skripty jsou soubory s příponou **.ps1* a protože psaní bloků těchto skriptů bude zmíněno až v kapitole 3, představme na tomto místě spuštění skriptu `pokus.ps1` s obsahem cmdletem `Get-Process`, který vypíše všechny běžící procesy v systému.

V kapitole 2.2 bylo čerpáno z [1], [2], [3], [4], [5], [6].

2.3 Roura

Mechanismus roury je v prostředí operačních systému Windows i Unix známý celou řadu let. Poprvé byl použit v 70. letech v prostředí Unix v raném období vývoje tohoto systému. Za jeho vznikem stojí Douglas McIlroy, jehož myšlenka byla vytvářet malé jednoúčelové prográmky a postupně je přes rouru spojovat do větších celků. Microsoft rouru poprvé využil v prostředí MS-DOS. Typickým příkladem je umožnění stránkování dlouhých výpisů:

```
dir | more
```

Z uvedeného příkladu je patrné, že se jedná o postupné použití dvou (případně i více) programů, které si předávají data právě pomocí této roury. Textová sekvence vykonaná příkazem `dir` (výpis obsahu adresáře) je předána příkazu `more`, který provede uspořádání výstupu. Jak je v příkladu též uvedeno, jako spojovací znak se používá „ | “. Zřetězení těchto programů a příkazů za sebe je možné v neomezeném počtu, což nabízí velký potenciál. Použití mechanismu roury pro komunikaci mezi programy je v některých případech nutné, neboť se najdou cmdlety, jejichž použití bez roury nenachází uplatnění.

2.3.1 Textová roura

V tomto případě vycházíme z předpokladu, že data jsou do roury i z ní zasílána formou textových řetězců. Jedná se tedy o jakousi sekvenci dat (písmena, číslice a další

znaky), která jsou chápána na obou stranách roury stejně. Možnosti roury se odvíjí od schopností jednotlivých programů, u textové roury je prvotní možností využití prohledávání řetězců.

V PowerShellu jsou data do roury zasílána ve formě objektů. Díky schopnosti PowerShellu provést za běhu konverzi s nimi můžeme pracovat jako s textovými řetězci. V případě, že externí program pracuje s daty přes rouru (je jedno v jakém směru) jako proud znaků, PowerShell se dokáže přizpůsobit.

2.3.2 Objektová roura

Jedná se o jednu z velkých předností PowerShellu. Při použití příkazů PowerShellu, typicky cmdletů, data vstupují do roury jako objekty. Pod tím si lze představit datovou strukturu, ve které jsou obsaženy opakující se položky. Celé seskupení těchto položek je pak nazýváno kolekcí. Jako ukázku použijeme už několikrát zmíněný cmdlet `Get-Process`:

```
Get-Process | Format-List
```

Tato příkazová sekvence na výstupu vypíše kolekci všech běžících procesů. Každý proces je zde definován jako jedna instance. Kolekce v rouře obsahuje počet instancí s pevnou vnitřní strukturou, odpovídající v tomto případě počtu běžících procesů. Každá z těchto instancí obsahuje svoje parametry, které jsou podle vlastností řádně pojmenovány. Při zkoumání parametrů není nutné brát ohled na jejich pořadí. Použití lze snadno ukázat na následujícím příkladu:

```
Get-Process -Id 5384
```

Id je v tomto případě vlastnost, která se váže k objektu procesu. Tento fakt nutno zdůraznit, neboť jiný objekt (například služba), už bude mít vlastnosti jiné. Provedení příkazu vrátí výpis s příslušným procesem:

```
Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    565      50   514148     532864   728 1 532,26   5384 firefox
```

Z příkladu lze vyčíst, že objektová roura umožňuje jakýsi univerzální přístup k datům. Po jednoduché ukázce je na místě zmínit, jak získat seznam všech vlastností,

které má daný objekt v rouře k dispozici. Toho docílíme použitím cmdlet `Get-Member`, příklad pro představu:

```
Get-Process | Get-Member
```

Tento zápis vrátí popis objektu procesu, vypadat může takto:

```
   TypeName: System.Diagnostics.Process

Name             MemberType     Definition
----             ----------     ----------
Handles          AliasProperty  Handles = Handlecount
Name             AliasProperty  Name = ProcessName
.....................................................
Dispose          Method         System.Void Dispose()
Equals           Method         bool Equals(System.Object obj)
.....................................................
Id               Property       System.Int32 Id {get;}
MachineName      Property       System.String MachineName {get;}
```

Z tohoto výpisu je zajímavá část s položkami typu Property, neboť se jedná právě o vlastnosti, na které se chceme u objektů procesů dotazovat. Nyní je zřejmé, jak se získá seznam vlastností a je na místě upřesnit, jak konkrétní vlastnost použít. Na použití konkrétní vlastnosti objektu v rouře se používá proměnná se speciálním zápisem spojená ze znaků dolaru a podtržítka ‚$_‘. K tomuto speciální znaku se za tečku přidá název konkrétní vlastnosti. Pro představu je tu jednoduchý příklad, který vypíše všechny běžící služby:

```
Get-Service | Where-Object {$_.status -eq "running"}
```

Přes mechanismus roury lze výstupy dále různě formátovat dle potřeby pomocí cmdletů `Format-List`, `Format-Wide`, `Format-Custom`, `Format-Table`. K třídění výstupů lze použít další objektové cmdlety `Foreach-Object`, `Where-Object`. První jmenovaný provede určitou operaci se všemi objekty v rouře, druhý jen s těmi, které splňují danou podmínku.

V kapitole 2.3 bylo čerpáno z [1], [3].

2.4 Uživatelské profily

V PowerShellu je využito tzv. uživatelských profilů, neboli „user profile“. Pod uživatelským profilem si lze představit skript, který je spuštěn při každém spuštění PowerShellu. Jeho úkolem je nastavit prostředí PowerShellu pro konkrétního uživatele nebo v opačném případě načíst výchozí konfiguraci. Všechny parametry určitého nastavení jsou uloženy v souboru **profile**. Při startu PowerShellu je možné zpracovat nejvýše čtyři tyto soubory.

Profil konkrétního uživatele je uložen v Dokumentech ve složce ‚WindowsPowerShell‘. Globální profil je pak možné nalézt v systémovém adresáři: ‚Windows\System32\WindowsPowerShell\‘.

Existence profilu se ověří takto:

```
test-path -path $profile
```

Profil lze vytvářet či dále modifikovat otevřením skriptu příkazem:

```
notepad $profile
```

V kapitole 2.4 bylo čerpáno z [7].

2.5 Aliasy

Powershell umožňuje pro všechny své příkazy podporu aliasů. Ty mohou být definovány v rámci systému nebo v rámci profilu uživatele. Alias je pouze jiný název pro příkaz. Pro představu lze cmdlet `Get-ChildItem` zapsat zkráceně příkazem `dir`. Seznam aliasů pro nejvíce používané cmdlety je uveden v tabulce č. 2.1. Seznam všech dostupných aliasů lze vypsat pomocí cmdletu `Get-Alias`. V PowerShellu jde spousta věcí řešit oběma směry, je tomu tak i v případě aliasů, následující příkaz zjistí všechny aliasy pro jeden konkrétní cmdlet:

```
Get-Alias | Where-Object {$_.Definition –like „Where-Object“} |
Format-Table Definition, Name
```

Tento postup lze samozřejmě zapsat i jinak, záměrně bude nyní uveden stejný příkaz zapsaný čistě přes aliasy, který dokáže i jejich negativní stránku, a naznačí, že s aliasy to není dobré přehánět.

```
gal | ? {$_.Definition -like 'where*'} | ft n*, def*
```

Aliasy je možné i dále vytvářet cmdletem `Set-Alias`. Ten nastaví alias k původnímu příkazu. V tomto případě se nemusí jednat pouze o cmdlet, ale i nějaký skript, případně

cestu ke spustitelnému (*.exe) souboru. Možností využití cmdletu `Set-Alias` lze získat takto: `Get-Help Set-Alias –examples`

V kapitole 2.5 bylo čerpáno z [4], [8].

2.6 Práce s daty

PowerShell podobně jako jiné programovací nebo skriptovací jazyky disponuje bohatou výbavou datových struktur. Vazba s běhovým prostředí .NET Framework umožňuje z tohoto hlediska stejné možnosti, které mají všechny ostatní jazyky založené na .NET.

2.6.1 Proměnné

Proměnná je základním úložným místem pro data. Podobně jako u jiných skriptovacích jazyků je uvozovacím znakem dolar ‚$'. Za ním následuje libovolný řetězec. Přiřazení hodnoty k proměnné se provádí klasicky znakem ‚='. Obsah proměnné lze získat zapsáním a předáním interpretu. Do proměnné lze přiřadit vesměs cokoliv, ať už třeba existující proměnná nebo i cmdlet. PowerShell sám určí datový typ podle hodnoty proměnné. U celočíselného přiřazení rozpozná, že se jedná o Integer, zápis v uvozovkách vyhodnotí jako String. Není-li si uživatel jistý, jaký datový typ PowerShell jeho proměnné přiřadil, může toto snadno ověřit následujícím zápisem: `Write-Host $MocninaDvou.GetType().Name`. Důležitou poznámkou, je skutečnost, že datový typ proměnné PowerShell dokáže měnit. Pokud je do proměnné zapsána celočíselná proměnná a v dalším přiřazení text v uvozovkách, PowerShell změní datový typ proměnné z Integeru na String. Pokud chce uživatel datový typ určit dopředu, může tak učinit explicitní deklarací současně s přiražením. Např. takto: `[int] $pozice = 125`. PowerShell pak do této proměnné nedovolí přiřadit například textový řetězec a při takovém pokusu vypíše chybové hlášení.

2.6.2 Datové typy

PowerShell přebírá datové typy od platformy .NET Framework, která jich nabízí široké množství. Přehled těch nejvíce používaných je uveden v tabulce 3.1.

označení	datový typ
[datetime]	datum nebo čas
[string]	řetězec znaků
[char]	jednotlivý znak
[double]	desetinné číslo s dvojnásobnou přesností
[single]	desetinné číslo se základní přesností
[int]	celé číslo reprezentováno 32 bity
[array]	pole hodnot
[xml]	xml objekt
[hashtable]	hashtable objekt
[wmi]	instance nebo kolekce objektů WMI
[adsi]	objekt získaný rozhraním Active Directory Services (ADSI)
[wmiclass]	třída objektového modelu WMI
[boolean]	logická hodnota dle Booleovy algebry; jeden bit; ano/ne;

Tabulka 3.1. *Přehled datových typů*

V kapitole 2.6 bylo čerpáno z [1], [4], [5], [9], [10].

2.7 Prostředí

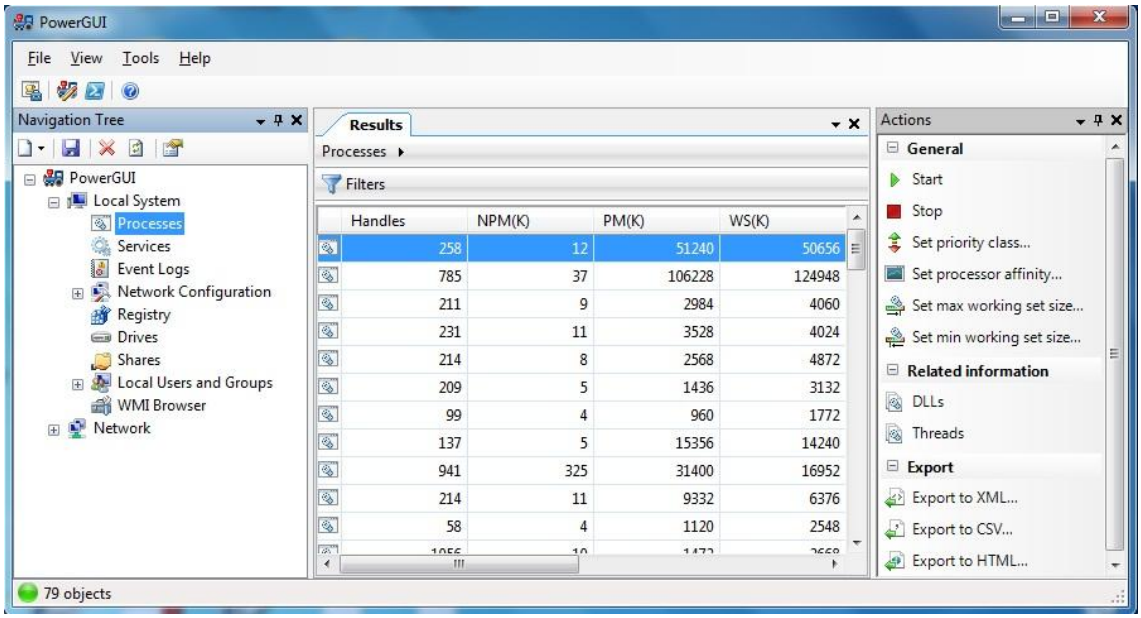
Používání PowerShellu i vývoj skriptů je možný v různých prostředích, jejich přehled nabídne tato podkapitola. Základní spuštění PowerShellu bylo již zmíněno v kapitole 2.2. Nyní budou představena vývojová prostředí pro tvorbu skriptů.

2.7.1 PowerShell ISE

Toto prostředí je novinkou pro verzi 2. Je tedy k dispozici až v operačních systémech Windows 7 nebo Windows Server 2008 R2. Jak vyplývá ze zkratky názvu ISE (Integrated Scripting Environment) jedná se o propojení skript editoru a konzole. Mimo tyto dvě součásti obsahuje ještě panel pro výpis výstupních dat. Konzole i editor umožňují podobně jako jiná pokročilá vývojová prostředí možnost našeptávání doplnění kódu pomocí tabulátoru. Podporována je též práce se schránkou. Velkou výhodou je i možnost otevření nové instance PowerShellu v další záložce. Vývojové prostředí umožňuje další standardní prvky jako debugování nebo krokování.

2.7.2 PowerGui

Jedná se o jedno z nejlepších rozšíření, které bylo k PowerShellu vyvinuto. Nabízí k dispozici dvě části. Ta první je klasická, obdobná jako u PowerShell ISE. Jedná se o klasický editor skriptů s možností ladění. Vynikající možností je však druhá část. Jde o grafické správcovské rozhraní postavené na PowerShellu. Všechny zadané operace jsou převedeny na kód v PowerShellu a v něm následně provedeny. Software je k dispozici volně, je podporován firmou Quest Software, která přináší hned několik možností rozšíření pro PowerShell. Ukázka spuštěného prostředí je na obrázku 2.1.



Obrázek 2.1: *Prostředí PowerGUI*

V kapitole 2.7 bylo čerpáno z [1], [4], [9], [10].

2.8 Rozšíření a doplňky

Protože je PowerShell poměrně nový produkt a přestože nabízí řadu možností, jeho základní funkcionalita může správci výpočetního systému působit poněkud skromně. Ceněnou vlastností je tak jeho modularita, díky níž se může dále rozšiřovat. Nejčastěji používané doplňky budou představeny v této kapitole.

2.8.1 PowerShell Comunity Extensions

Součástí balíku, který je k dispozici zdarma, je široká škála doplňků. Zahrnuje nové cmdlety, z nichž nejzajímavější je `Send-SmtpMail`. Přehled všech nových cmdletů lze získat příkazem `gcmpscx`. Všechny přidané cmdlety disponují kvalitní nápovědou. Dále jsou k dispozici předdefinované funkce a také součásti typu provider, které umožňují přístup k dalším zdrojům operačního systému. Instalace doplňku je snadná přes klasický instalátor.

2.8.2 ActiveRoles Management Shell for Active Directory

Jak již název napovídá, tento balík má velký význam z hlediska správy adresářových služeb Active Directory. Samotný PowerShell tuto možnost přináší až ve verzi 2, tento doplněk se tady váže zejména k první verzi PowerShellu. Stejně jako PowerGui pochází z dílny společnosti Quest Software. Přináší řadu nových cmdletů, které umožňují přímé propojení s Active Directory. Díky tomu je možné mimo jiné manipulovat s uživatelskými účty nebo vytvářet, mazat či modifikovat různé objekty. Stejně jako předchozí doplněk je i tento k dispozici zdarma, včetně kvalitní přehledné dokumentace.

2.8.3 NetCmdlets

První komerční verze (pro stanici stojí 99$, pro server pak 299$, zkušební verze zdarma – červen 2011) přináší rozšíření v oblasti síťových funkcí a patří mezi nejlepší doplňky, z hlediska faktu, že na síťové funkce nebylo ve výchozí podobě PowerShellu myšleno. Stejně jako předchozí balík nabízí správu Active Directory, dále možnost vzdáleného přístupu do příkazové konzole přes technologii SSH a velmi dobrou novinkou je též podpora protokolu SNMP v3. Tím výčet novinek nekončí, balík přináší i možnost práce s e-mailem, podporu přenosu souborů přes FTP nebo možnost monitorování počítačové sítě. Umí také komprimovat soubory, příjemnou novinkou je i RSS čtečka.

2.8.4 PowerGadgets

I tento balík je komerční, je dostupný za 99$ (červen 2011). K dispozici je také i zkušební verze zdarma. Jde o určitý paradox. Zatímco doposud byl PowerShell zmiňován jako čistě konzolové rozhraní, balík rozšíření PowerGadgets přichází s grafickým rozšířením. Tvůrci vhodně využili možnosti objektové roury a s její pomocí přesměrovali data na komponentu, která z nich zpracuje grafický výstup v podobě grafu, histogramu nebo jiných výstupních prvků.

2.8.5 Windows System Modules

Myšlenka modulů přichází s PowerShellem verze 2. Ve verzi 1 byla rozšíření k dispozici ve formě ‚ddl' souborů, takzvaných snap-inů. Moduly umožňují jinou a snazší formu rozšíření jen za využití textových nebo binárních souborů.

Jak již je v PowerShellu zvykem, i u práce s moduly se použije cmdlet s příznačným označením `Get-Module`. Tento cmdlet umožňuje vypsat instalované či dostupné moduly. Dostupné moduly jsou vypsány na základě adresářů s moduly, které jsou definovány v proměnném prostředí ‚**$env:PSModulePath**'. Tyto lze pak dále importovat. K jejich případnému přidávání nebo odebírání poslouží další dva cmdlety `Import-Module` a `Remove-Module`.

Moduly je samozřejmě možné vytvářet i vlastní. K zjednodušení této práce existuje zajímavý nástroj Module Management, který je k dispozici jako rozšíření vývojového prostředí Power GUI (uvedeného v podkapitole 2.7.2).

Nejnovější serverová verze Windows Server 2008 R2 už disponuje možností přidání řady zajímavých modulů, které jsou k dispozici ve výchozím adresáři PowerShellu a je možné je hned začít používat. Mezi tyto moduly patří mimo jiné Active Directory PowerShell, podpora pro přenos souborů BITS nebo ovládání služby IIS. Tyto moduly lze do systému přidat jak z grafického rozhraní operačního systému, tak z konzole tímto příkazem:

```
powershell -importsystemmodules
```

V kapitole 2.8 bylo čerpáno z [1], [3], [4], [9], [11], [13].

3 Přístup k rozhraním

Tato kapitola bude zaměřena na možnosti PowerShellu pro správu pracovních stanic přes nástroj Active Directory a služby WMI.

3.1 Active Directory

Firma Microsoft vyvinula pro implementaci adresářových služeb LDAP (Lightweight Directory Access protokol) nástroj Active Directory. Ten je použit v prostředí serverového operačního systému Microsoft Windows. Správcům počítačových sítí umožňuje jednoduše nastavovat politiky zásad, na několik počítačů najednou instalovat programy nebo instalovat kritické aktualizace v celé organizační struktuře. Všechna nastavení a informace jsou ukládány v centrální databázi.

Active Directory je rozšiřitelnou adresářovou službou, která umožňuje efektivní uspořádání síťových prostředků. Skupiny uživatelů, počítačů a domén jsou organizovány do domén, jejichž činnost je založena na standardních síťových protokolech. Struktura počítačové sítě z hlediska jmenovaných prvků je pak jednoznačně definována. K činnosti Active Directory je potřebná instalace služby DNS (Domain Name System), jejíž hlavní činností je vzájemný převod doménových jmen a IP adres uzlů počítačové sítě.

3.1.1 Rozhraní ADSI

Active Directory Service Interfaces je sada rozhraní COM (Component Object Model) používané k přístupu k funkcím adresářových služeb. Používá se v distribuovaných výpočetních prostředí k jednotnému nastavení adresářové služby pro správu síťových zdrojů. Administrátoři mohou ovládat adresářové služby bez ohledu na síťové prostředí. Umožňuje běžné úkoly správy jako je přidávání nových uživatelů, správa tiskáren a umístění zdrojů v distribuovaném výpočetním prostředí. Správci sítě mohou používat ADSI k automatizaci běžných úkolů, jako je přidání uživatelů a skupin, správa tiskáren a nastavení oprávnění u síťových zdrojů. To vše je dosaženo se zachováním stručné a pochopitelné formy zápisu. Rozhraní ADSI nabídlo vedle správy Active Directory ještě další možnosti, z nichž nejzajímavější je administrace běžících služeb.

3.1.2 Vnější struktura Active Directory

Součástí Active Directory jsou fyzické i logické síťové struktury.

Fyzické

Podsítě – skupina, která obsahuje určitý rozsah IP adres a masky podsítě

Sítě – jedna nebo více podsítí, slouží ke konfiguraci přístupu k adresářové službě

Logické

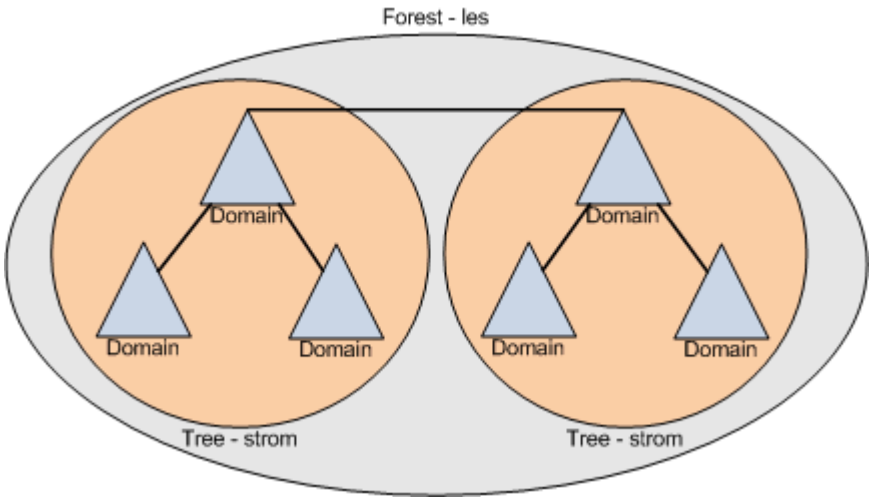
Domény – skupiny počítačů, sdílející společnou adresářovou databázi

Organizační jednotky – podskupiny (nejnižší organizační prvek) domén, často odpovídající řídící nebo obchodní struktuře organizace

Stromy domén – jedna či více domén, které sdílejí společné adresářové informace

Lesy domén – jeden nebo více stromů domén, které sdílejí společné adresářové informace

Příklad doménové struktury je znázorněn na obrázku 3.1.



Obrázek 3.1: *Doménová struktura*

3.1.2 Vnitřní struktura Active Directory

Active Directory zpřístupňuje uživatelům a počítačům svá data prostřednictvím úložišť dat a globálních katalogů. Úložiště dat ovlivňuje většinu úkolů Active Directory. Globální katalogy však také mají svůj velký význam, jelikož jsou využívány

při přihlašování a hledání informací. K datům Active Directory je přistupováno pomocí protokolů sloužících k přístupu k adresářové struktuře.

3.1.3 Přístup z PowerShellu

Připojit se k Active Directory je obecně možné několika způsoby. Nejsnadněji je to možné přes rozšiřující balík od firmy Quest, zmíněný v kapitole 2.8.2. Univerzálním prostředkem je zmíněné rozhraní ADSI, které umožňuje nejen automatizované postupy, ale také hromadné operace většího rozsahu.

Princip připojení z PowerShellu je prakticky obdobný jako u přístupu k WMI rozhraní. Navíc je využito statických tříd .NET Frameworku.

Ukázka vytvoření nového uživatele v Active Directory (doména: ‚dusin.pokusny', organizační jednotka: ‚pokusna'):

```
$OU = [ADSI]'LDAP://ou=pokusna,dc=dusin,dc=pokusny'
$User = $OU.Create("user", "cn=Blaho")
$User.Put("samAccountName", "blahnikj")
$User.SetInfo()
```

V kapitole 3.1 bylo čerpáno z [9], [12], [14].

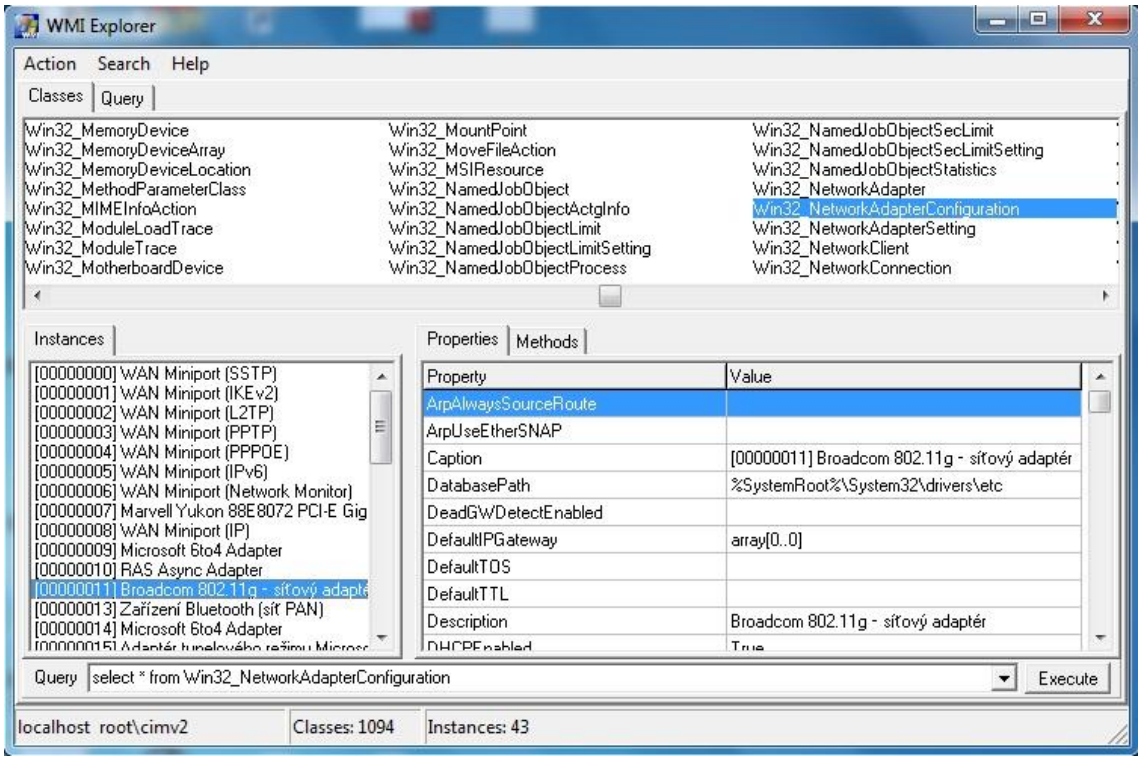
3.2 WMI (Windows Management Instrumentation)

Jedná se o univerzální prostředek pro pokročilou administraci Windows. Je využit například v nástrojích Microsoft Operations Manager či System Management Server. Jeho zpřístupnění ve skriptovacím prostředí pak posunulo možnosti nových monitorovacích a ovládacích rozhraní.

Služba WMI obsahuje úložiště objektů, které je databází jejich definic, a dále obsahuje nástroj WMI Object Manager. Jeho význam spočívá ve shromažďování objektů a manipulaci s nimi v úložišti. Také shromažďuje informace od zprostředkovatelů služby WMI. Ti zajišťují komunikaci mezi součástmi operačního systému a službou WMI, aplikacemi a dalšími systémy. Zprostředkovatel registru informace získává z registru, zprostředkovatel SNMP (Simple Network Management Protocol) naopak poskytuje údaje a události ze zařízení SNMP. Zprostředkovatelé poskytují informace o svých komponentách a případně mohou nabízet metody pro práci

s komponentami. Na obrázku 3.2. je uvedena ukázka získávání systémových informací pomocí programu WMI Explorer.

Využitím tohoto rozhraní se výrazně zvyšují možnosti sledování a správy systémů Windows bez nutnosti využití složitých prostředků.



Obrázek 3.2: *Spuštěný program WMI Explorer*

3.2.1 Získávání informací

Zde se již konkrétně budeme bavit o získávání informací pomocí PowerShellu. V tomto případě se opět ukáže velká výhoda svázání s běhovým prostředím .NET Framework, díky němuž je práce s WMI přirozená. Velkou výhodou je objektově orientované uspořádání WMI oblastí. Díky tomu je možné příchozí data použít díky mechanizmu roury. K získání informací z rozhraní WMI používá PowerShell cmdlet `Get-WmiObject`. Za něj se pak jako parametr přidá požadovaná třída WMI rozhraní. Výpis pak lze různě sestavit podle představ, která konkrétní data jsou potřeba.

Příklad použití:

```
Get-WmiObject Win32_NetworkAdapterConfiguration
```

Tento zápis vypíše vlastnosti instalovaných síťových rozhraní.

Protože pro různé systémové oblasti jsou potřeba data na základě různých vlastností, hodí se získat seznam všech vlastností pro konkrétní třídu. Toho lze docílit využitím již z kapitoly 2.3.2 známého cmdletu `Get-Member` tímto zápisem:

```
Get-WmiObject Win32_NetworkAdapterConfiguration |
Get-Member -MemberType property
```

3.2.2 Volání akcí

Rozhraní WMI přináší i druhou zajímavou věc, kterou je volání metod nad objekty. Jako zajímavý příklad opět poslouží síťové adaptéry:

```
(Get-WmiObject -List | Where-Object -FilterScript {$_.Name -eq
"Win32_NetworkAdapterConfiguration"}).RenewDHCPLeaseAll()
```

Tento zápis přepne všechny síťové adaptéry do stavu klientů služby DHCP a získají tak dynamicky veškeré konfigurační údaje. V principu je použití stejné jako u získávání informací. I tentokrát je výchozím prvkem cmdlet `Get-WmiObject`. Seznam dostupných metod pro každou WMI třídu se opět snadno zjistí přes cmdlet `Get-Member`.

V kapitole 3.2 bylo čerpáno z [1], [4], [15], [16].

4 Vybrané části PowerShellu

4.1 Logy

Při práci s logy umožňuje PowerShell zpracování celků (souborů logů) i práci s jednotlivými událostmi uvnitř logů. Základním cmdlet pro obě tyto činnosti se jmenuje `Get-EventLog`. Jeho výstup je opět možné klasicky upravovat a třídit podle požadavků administrátora. Tímto způsobem lze vybírat jen určité logy nebo je seskupovat podle typu. V případě získávání jednotlivých událostí je ke zmíněnému cmdletu přidán jako parametr název konkrétního logu. V případě, že název zadán nebude, PowerShell se na něj sám dotáže. Důležitým přepínačem cmdletu pro výpis logů je klíčové slovíčko ‚Newest'. S jeho využitím se získá potřebný počet nejnovějších záznamů událostí.

Ukázky použití:

```
Get-EventLog system -newest 3
```

Tento jednoduchý zápis vypíše tři nejnovější události v systémovém logu.

```
Get-EventLog system | Where-Object {[DateTime]::Now.AddDays(-7)
-le $_.TimeWritten} | Group-Object eventid |
Sort-Object count -descending | ft count,name –AutoSize
```

Druhý zápis je výrazně složitější. Vypíše události za posledních 7 dní ze systémového logu, sjednotí je podle ID a vypíše sestupně seřazené.

4.2 Služby

Dříve se správa služeb v systémech Windows prováděla přes aplikaci sc.exe. PowerShell přináší pro tuto činnost podstatně širší možnosti, které se mu stejně jako v předešlých případech naskytují využitím objektových aspektů a to zejména při práci s mechanizmem roury. K práci se službami se používá cmdlet `Get-Service`. K němu se zpravidla jako první parametr (určen pozicí – viz kapitola 2.2.) používá název služby. Opět je možné výpis služeb dále upravovat podle potřeb, vypisovat služby obsahující v názvu nějaký řetězec, případně služby, které jsou v nějakém konkrétním stavu atd. Zajímavostí je výpis služeb podle závislostí a to oběma směry. Lze vypsat služby závislé na konkrétní službě i opačně, tedy služby, na nichž závisí jedna konkrétní.

Příklad – výpis aktuálně běžících služeb:

```
Get-Service | Where-Object {$_.status -eq "running"}
```

Cmdlet pro výpis služeb pomůže zjistit celou řadu informací, přesto však existuje ještě lepší řešení pro práci se službami v PowerShellu. Vrátíme se do kapitoly 3.2.1. k získávání informací z WMI rozhraní. Nabízí se vhodné využití třídy ‚Win32_Service', která o službách přináší ještě více informací, než do teď zmíněný cmdlet. Důležitou vlastností je, zda je služba spouštěna automaticky nebo manuálně. Pro příklad služby spouštěné manuálně zjistíme přes WMI takto:

```
Get-WmiObject Win32_Service | Where-Object {$_.StartMode -match
"auto*" -and $_.State –notmatch "run*"} | Format-Table -auto
```

Cmdlety `Get-WmiObject` a `Get-Service` je možné díky objektům služeb při správě různě kombinovat. Nad službami se běžně provádějí operace spuštění, zastavení a restartu služby, tyto činnosti jsou obsluhovány cmdlety: `Start-Service`, `Stop-Service` a `Restart-Service`.

Příklad spuštění služby:

```
Get-WmiObject Win32_Service | Where-Object {$_.Name -eq
" Apache2.2 "} | Start-Service
```

4.3 Procesy

I ve správě procesů poskytuje PowerShell významně lepší možnosti než jeho předchůdci. Základem práce s procesy je cmdlet `Get-Process`. Ten zadaný bez parametrů vypíše všechny běžící procesy. Jako první parametr, opět pevně definovaný pozicí, je možné uvést název procesu nebo jeho fragment (část názvu). Jako je již v PowerShellu zvykem, i zde lze snadno procesy zpracovávat objektově. Pracovní cmdlet nabízí spoustu velmi zajímavých vlastností, zejména pro práci s pamětí procesů, ale i například informaci o výrobci modulu. Pro ukázku, tímto zápisem se zajistí výpis výrobců, jejichž software právě v počítači běží:

```
Get-Process | Sort-Object company -Unique |
Format-Table company -autosize
```

V kapitole 4 bylo čerpáno z [1], [3], [4], [5], [9], [17].

5 Analýza vhodných činnosti pro správu

Na základě teoretických poznatků je vhodné vybrat činnosti, které jsou potřebné pro usnadnění správy domény na KIV. Díky PowerShellu je možné spravovat rozsáhlé oblasti systému. Nyní je potřeba vybrat ty nejvíce potřebné pro potřeby KIV a vytvořit z nich jeden administrátorský systém.

5.1 Zvolené činnosti

Hlavní činnosti:

- práce s uživateli adresářové služby Active Directory
- získávání informací na základě WMI objektů
- správa služeb a procesů
- podpora vytváření a kontroly účtů MSDN

Z uvedených hlavních činností mohou vycházet některé další. Například vytvoření mechanizmu kontroly studentů, zda ještě studují na KIV, či mají některý předmět z KIV zapsaný. Active Directory je z hlediska správy domén uživatelských účtu klíčový nástroj a přímá interakce s ním může snadno vést k chybě s nepříjemnými následky (smazání objektu, úprava vlastností jiného objektu a jiné). Nabízí se možnost využít PowerShell pro výrobu skriptů, které umožňují vytváření uživatelských účtů a eliminují možnosti vzniku některých chyb.

5.2 Využití webových služeb

K zajištění zvolených činností je nutné zajistit přístup ke studijní agendě IS/STAG. Z ní je třeba získat informace o studentech, předmětech a případně dalších informacích. Získání těchto informací je možné s využitím webových služeb, jak je naznačeno na obrázku 5.1.

Obrázek 5.1: *spolupráce s IS-STAG*

Webové služby vrací získaná data v různých formátech. Pro použití do systému využívajícího pro práci s uživateli PowerShell je nejvhodnější zvolit výstupní formát dat v XML. PowerShell totiž umí s XML soubory pracovat a konkrétní potřebná data analyzovat.

Webové služby IS-STAG používají dva základní standardy:

- **SOAP** Využívá pro přenos informací mezi stroji zavedených protokolů. Tunelování (zapouzdření spojení) zde funguje tak, že dojde k zabalení SOAP zprávy do zprávy protokolu, kterému firewall důvěřuje.

- **REST** Metoda REST využívá protokolu HTTP, konkrétně jeho metod: POST, PUT, DELETE a takto dochází k předávání dat službě.

Webové služby jsou umístěny na serveru: **https://stag-ws.zcu.cz/ws/services**. Pro potřeby návrhu systému byl pro svoji jednoduchost zvolen standard REST. Konkrétně dvě jeho webové služby:

- /rest/student
- /rest/predmety

Aktuální seznam webových služeb IS/STAG pro ZČU je uveden zde:
https://stag-demo.zcu.cz/ws/help/list

5.3 Získaná data

Každá z uvedených služeb v kapitole 5.2 ještě obsahuje operace pro přístup k různým konkrétním datům. Ukázka části výstupního XML souboru pro použití operace getStudentiByPredmet:

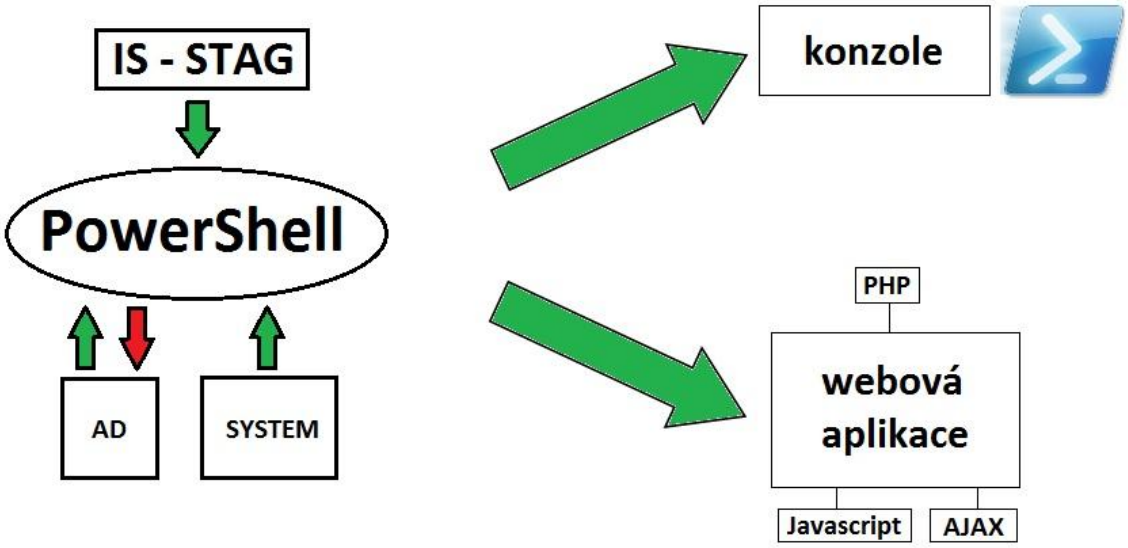
```
<ns1:getStudentiByPredmetResponse xmlns:ns1="http://stag-
ws.zcu.cz/">
  <studentiPredmetu>
    <studentPredmetu>
      <osCislo>A09N0025P</osCislo>
      <jmeno>Miroslav</jmeno>
      <prijmeni>BERAN</prijmeni>
      <titulPred>Bc.</titulPred>
      <stav>S</stav>
      <userName>berry</userName>
      <stprIdno>762</stprIdno>
      <nazevSp>Inženýrská informatika</nazevSp>
      <fakultaSp>FAV</fakultaSp>
      <kodSp>N3902</kodSp>
      ..........................
    </studentPredmetu>
  </studentiPredmetu>
</ns1:getStudentiByPredmetResponse>
```

S takto získaným souborem už dokáže PowerShell pracovat. Na XML soubor hledí jako na objekt a snadno získá položky konkrétního elementu, v tomto případě `<studentPredmetu>`. Jako příklad poslouží následující fragment kódu, který do proměnné ‚xml' uloží obsah souboru získaného webovou službou a k získání údajů jména a příjmení studenta a jejich následného použití použije dva cykly. První cyklus projde všechny elementy `<studentPredmetu>` v XML souboru a ten druhý v patřičném elementu projde všechny položky a uloží potřebná data.

```
$xml = [xml] (get-content students.xml)
ForEach($node in $xml.GetElementsByTagName("studentPredmetu")){
  ForEach($item in $node){
    $jmeno= $item.jmeno
    $prijmeni= $item.prijmeni
    $login= $item.userName
    #akce s pouzitim jmena, prameni a loginu
  }
}
```

6 Návrh systému

Pro činnosti zvolené v kapitole 5 je potřeba navrhnout systém, který je bude realizovat. Aplikační logiku budou provádět navržené skripty v PowerShellu, které je možno spouštět z konzole a dále navrhnout uživatelsky přívětivé prostředí, ideálně formou webové aplikace. Architektura systému popisující jeho funkci je naznačena na obrázku 6.1.



Obrázek 6.1: *Architektura navrhovaného systému*

6.1 Spouštění z konzole

V tomto případě lze použít cmdlety nebo ručně psané skripty uložené v souborech **.ps1*. Skripty je možné spustit z příkazového prostředí PowerShellu standardním způsobem tímto zápisem:

```
skript.ps1
```

Nebo z klasického příkazového prostředí Windows **‚cmd‘** tímto zápisem:

```
powershell.exe -command script.ps1
```

PowerShell skripty z bezpečnostních důvodů nelze samovolně spouštět a bezpečnostní politika je součástí jeho základních nastavení. Při prvním použití PowerShellu je politika nastavena na přísný režim a žádné skripty není možné spustit.

Současné nastavení bezpečností politiky se získá spuštěním cmdletu `Get-ExecutionPolicy`. Bezpečností politika má následující možnosti nastavení:

- **Restricted** žádný skript nelze spustit
- **AllSigned** lze spustit pouze důvěryhodné skripty
- **RemoteSigned** nebrání činnosti lokálních zdrojů, ale zabraňuje spouštění vzdáleně uložených skriptů
- **Unrestricted** úplné odblokování spouštění skriptů

Zápis pro změnu bezpečnostní politiky:

```
Set-ExecutionPolicy -ExecutionPolicy unrestricted
```

6.2 Webová aplikace

Druhou variantou, uživatelsky přívětivější, je ovládání PowerShellu z webového rozhraní. V systému s webovým rozhraním bude mít uživatel k dispozici menu, ve kterém si zvolí oblast činností správy systému. Pro každou oblast je k dispozici vlastní panel, který umožňuje provádět operace odpovídající příslušné oblasti. Oblasti budou dále popsány v pořadí, v jakém jdou za sebou v menu realizované webové aplikace. Jejich přehled a stručný popis také znázorňuje tabulka 6.1.

Modul oblasti	Popis činnosti
Generování skriptů	Slouží ke generování skriptů pro vytvoření uživatelských kont studentů v Active Directory nebo v SŘBD Oracle.
Správa Active Directory	Přímo komunikuje s Active Directory, vytváří uživatelská konta a členství ve skupinách na základě informací o studentech a předmětech získaných z IS/STAG.
MSDN	Umožňuje přidávat studenty podle předmětů do systému MSDN. Dokáže i získávat seznam studentů, kteří mají do MSDN přístup, ale již nejsou studenty.
WMI objekty	Využívá rozhraní WMI k přístupu k systémovým informacím.
Služby a procesy	Zprostředkovává informace o běžících službách a procesech a umožňuje je spravovat (měnit různá nastavení, spouštět, ukončovat apod.)

Tabulka 6.1: *Přehled modulů webové aplikace*

6.2.1 Generování skriptů pro vytváření uživatelských kont

Cílem této činnosti je ovládání Active Directory s ohledem na bezpečnost. PowerShell nepřistupuje k Active Directory přímo, ale pouze vyrábí skripty, které slouží k vytváření uživatelských kont. V tomto modulu je podobným způsobem vyráběn i skript pro vytvoření kont v systému řízení báze dat Oracle potřebných pro předmět KIV/DB2.

Skript pro vytváření uživatelských účtu v Active Directory (nejedná se o skript psaný v PowerShellu, ale klasický skript ‚**bat**' v prostředí Windows) byl doposud vyráběn složitějším ručním mechanismem a díky PowerShellu bude vytvořen velmi snadno. Nabízí se otázka, proč řešit tento problém, když PowerShell sám zvládne Active Directory ovládat. Tato část návrhu systému má však zachovat dosavadní osvědčený režim zakládání uživatelských kont, pro případ, kdyby systém nebyl nasazen přímo na produkčním serveru.

Vstupními daty tohoto modulu webové aplikace jsou zkratky katedry a předmětu. V případě skriptu použitelného pro Active Directory je potřeba ještě uvést rok studia. Nesmí chybět ani volba typu skriptu mezi vytvářením uživatelských kont v Active Directory (AD) a SŘBD Oracle (DB). Na základě těchto údajů jsou pak patřičné skripty vygenerovány. Výstupní okna jsou dvě. Jedno pro standardní a druhé pro chybový výstup. Do chybového výstupu jsou zahrnuti studenti, kteří buď nestudují, nebo byl v jejich datech zaznamenán nějaký problém, zpravidla se občas objeví chybějící login studenta.

Ukázky generovaných skriptů:

```
call student.bat SPOS-students-11 mbarta "Milan BARTA"
call student.bat SPOS-students-11 wincent "Vaclav BELE"
call student.bat SPOS-students-11 quantim "Petr CERNOHOUZ"
call student.bat SPOS-students-11 cizam "Martin CIZEK"
call student.bat SPOS-students-11 dyrczyk "Jan DYRCZYK"
call student.bat SPOS-students-11 pfilo "Pavel FILO"
```

skript pro vytváření uživatelských účtů předmětu KIV/SPOS v Active Directory

```
create user diggy identified by „A10N0029P“
default tablespace users
temporary tablespace temp
profile default;
grant  essou,  essource to diggy;

create user jbartova identified by „A09N0006K“
default tablespace users
temporary tablespace temp
profile default;
grant  essou,  essource to jbartova;
```

skript pro vytváření kont pro přístup do SŘBD Oracle pro studenty předmětu KIV/DB2

Uvedené skripty se dají snadno z výstupního okna uložit do souboru, který pak lze použít jako skript ‚**bat**‘ v prostředí Windows. Chybová hlášení jsou vypsána do okna s error logem. Klasickým chybovým hlášením může být: „Student A10N0038P nema platny login“. Chybový log má pouze informativní význam a není třeba s ním dále nijak pracovat.

Aplikační logiku tohoto webového modulu tvoří celkem tři PowerShell skripty. Dva z nich na základě údajů získaných webovou službou, která je spouštěna z prostředí PowerShellu generují výstupní skripty a zbylý jeden generuje chybový log. Skript pro vytváření účtu v Active Directory musí jména a příjmení studentů obsahovat bez diakritiky jako znaky ASCII. PowerShell skript proto obsahuje funkci pro převod znaků, která vypadá takto:

```
# funkce na prevedeni do ASCII
function cesky ([string]$text)
{
$sdi = "áäčďéěíĺľňóô öřšťúů üýřžÁÄČĎÉĚÍĹĽŇÓÔ ÖŘŠŤÚŮ ÜÝŘŽ"
$bezdi = "aacdeeillnoo orstuu uyrzAACDEEILLNOO ORSTUU UYRZ"
$chars = $text.ToCharArray()
$newtext = ""

foreach ($char in $chars) {
      $index = $sdi.IndexOf($char)
      if ($index -gt -1)
            {
                  $newchar = $bezdi.Substring($index,1)
                  $newtext += $newchar
            }
            else
            {
                  $newtext += $char
            }
      }
$newtext
}
```

Funkce je volána způsobem:

```
$jmeno = cesky $item.jmeno
```

V praxi to znamená, že jméno studenta je v podobě pole znaků procházeno cyklem a pokud je nalezen znak s diakritikou, je nahrazen svým ekvivalentem bez diakritiky. Na konci funkce vrátí novou podobu jména bez diakritiky. Ten samý princip je použit také při převodu příjmení studenta.

6.2.2 Správa Active Directory

Jedná se o modul, který obsluhuje přímé spojení mezi webovými službami IS-STAG a Active Directory. Jako vstupní data mu stačí zadat pouze zkratku předmětu a katedry, na základě kterých si pak stáhne s pomocí webové služby studenty konkrétního předmětu a pak také název organizační jednotky, kam chceme uživatelská konta umístit. Standardně je zvolena organizační jednotka KIV-Students. Díky technologii přístupu PowerShellu k Active Directory dokáže skript vytvořit uživatelské účty ve zvolené organizační jednotce s přiřazením do příslušných skupin, jejichž názvy jsou složeny ze zkratky předmětu a aktuálního studijního roku. Pokud již uživatelský účet studenta existuje, bude pouze přiřazen do členství uživatelské skupiny předmětu. Toto definuje následující fragment skriptu:

```
$maping = $login + "@ZCU.CZ"
$user = Get-QADUser $login
    if ($user)
    {
      Add-QADGroupMember -identity $GroupName -member $maping
    }
    else
    {
      New-QADUser –Name $maping -SamAccountName $login
      -FirstName $jmeno -LastName $prijmeni -DisplayName "$jmeno
      $prijmeni" –ParentContainer
      'OU=KIV-Students,DC=dusin,DC=pokusny'
      Add-QADGroupMember -identity $GroupName -member $maping
    }
```

Tento kód používá rozšiřující balík Quest. Uživatelé budou vytvořeni do organizační jednotky **KIV-Students**, jejich založená konta budou obsahovat základní údaje: jméno, příjmení a login. Heslo účtu bude mapováno z Kerbera pomocí programu **ksetup** (Kerberos setup). Jedná se o nástroj, který umožňuje konfiguraci účtů systémů Windows Server s možností použít údaje ze serveru Kerberos V5.

Mapování se zajistí tímto spuštěním z příkazového prostředí Windows:

```
ksetup /domain /mapuser %1@ZCU.CZ %1
```

Tento modul umí obsluhovat ještě jednu činnost a to systém kontroly, zda studenti s vytvořenými konty v Active Directory stále ještě studují. Kontrola funguje tak, že nejprve dojde ke stažení seznamu všech studentů se zapsanými předměty z katedry KIV a tento seznam je dále porovnáván se seznamem staženým z Active Directory, konkrétně organizační jednotky KIV-Students. Shody seznamu získaného ze STAGu se seznamem získaným z Active Directory jsou od tohoto odebrány a výsledný takto získaný seznam obsahuje studenty, kteří v současné době na KIV nestudují a jejichž uživatelské účty je možno buď dočasně omezit, nebo úplně odstranit.

6.2.3 MSDN

Tato část webové aplikace umožní přidávat studenty do systému správy studentských licencí MSDN. Systém přidávání je založen na vybírání studentů po oborech vyučovaných na KIV. Pro zvolený obor, ať už bakalářský nebo magisterský, je vybrána skupina předmětů, studovaných právě v rámci tohoto oboru. Nutnou podmínkou je, že tyto předměty musí být studovány na KIV, protože ty ostatní ztrácí pro použití v souvislosti s MSDN význam. K získání seznamu předmětů vyučovaných v rámci konkrétního oboru je opět využita webová služba IS-STAG vysvětlená v kapitole 5.2. Ze zmíněné skupiny předmětů je úptřeba vybrat pouze ty předměty, jejich studenty chceme do systému správy studentských licencí přidat. Po spuštění a provedení generování se vypíše seznam studentů ve formátu: login@students.zcu.cz, který slouží jako přihlašovací údaj do systému správy studentských aplikací. Nabízí se samozřejmě předpoklad, že jeden student může současně studovat více vybraných předmětů. I na toto je myšleno a ve skriptu PowerShellu je provedeno odstranění duplicitních záznamů. Kromě standardního výstupu je zde opět chybový log, do nějž budou vypsáni studenti, u nichž došlo k výskytu nesrovnalostí (neexistující login) a nemohou být pro MSDN použiti. Standardní výpis je možné uložit do souboru.

Tento modul je ovládán dvěma PowerShell skripty. První z nich získá seznam předmětů s využitím webové služby pro získání předmětů oboru. Nutno podotknout, že i zde je nutné pročistit duplicity, neboť webová služba vrací některé předměty v rámci jednoho studijního oboru dvakrát. Předměty jsou skriptem seřazeny v abecedním pořadí. Druhý skript obstarává vlastní generování. Oba skripty patří k rozsáhlejším a stojí za zmínku si jejich část ukázat. Následující fragment kódu

zajišťuje zmíněnou kontrolu duplicit u loginů studentů a je zde také vidět, že pokud login (userName) studenta neexistuje, je výpis přesměrován do chybového logu.

```
    ForEach($item in $node){
      if ($item.userName)
      {
        $vlozit = 1
        $login= $item.userName

        #kontrola duplicit
        ForEach($test in $loginArray){
          If ($test -eq $login) {
            $vlozit = 0
          }
        }
        #ulozeni loginu do seznamu
        If ($vlozit) {
          $loginArray += $login
        }
      }
      else
      {
        $out = "Student " + $item.osCislo + " predmetu " +
                $predmet + " nema platny login"
        $out
      }
    }
  }
```

Rozšiřující funkcí tohoto modulu je stejný princip jako u kontroly studentských účtů v kapitole 6.2.2. I zde je třeba kontrolovat, zda studenti s účty pro přístup do MSDN jsou ještě řádnými studenty. Tato kontrola je prakticky provedena shodně jako u Active Directory v předchozí kapitole. I zde jsou dva seznamy, mezi nimiž se vytváří rozdílový seznam. Administrátor si vybere seznam předmětů, pro které chce kontrolu provést a postupuje stejně, jako kdyby chtěl uživatele do systému MSDN přidat. Místo uložení tohoto seznamu do textového souboru však tentokrát zvolí možnost

porovnávacího formuláře. Jedná se o vstupní okno, kam musí být ručně vložen seznam současných uživatelů MSDN. Kontrola je pak provedena odstraněním shodných záznamů ze seznamu současných uživatelů MSDN.

6.2.4 WMI

Předposlední část webové aplikace využije rozhraní WMI pro poskytování informací. Modul využívá možnosti PowerShellu pro práci s WMI. Nabízí tak základní informace o systémových vlastnostech jako je verze BIOSu nebo informace o discích.

V případě BIOSu je vypsána jeho verze. Informace o discích jsou vypsány do přehledné tabulky, která poskytuje údaje o jednotkách, jmenovkách svazků, velikosti a volném místě.

6.2.5 Služby a procesy

Tento modul slouží ke správě služeb a procesů. Se službami i procesy je v PowerShellu zacházeno jako s objekty.

V této oblasti webové aplikace je umožněno služby zobrazit ve čtyřech různých kategoriích. V základní podobě jsou vypsány všechny služby. Dále je však možné je různě vytřídit na základě toho, zda jde o služby právě běžící či zastavené nebo o služby, které jsou spouštěné automaticky. Ve výpisu služeb je u názvu každé služby uvedeno několik údajů. Těmi statickými, které jen zobrazují informace o službě je stav služby a mód jejího spuštění (zda je spouštěna automaticky nebo ručně). Tyto informace lze pomocí správy služeb měnit. Službu lze spustit či zastavit. Je také možné změnit mód jejího spouštění z ručního na automatický či naopak. Informativní význam má také zobrazení závislostí pro danou službu. Činnost výpisu služeb a jejich ovládání je řízena přes jednoduché skripty psané v PowerShellu využívající informace z WMI z důvodu využití širšího okruhu informací jak bylo vysvětleno v kapitole 4 zabývající se službami a procesy v PowerShellu.

Správa procesů je vedena podobně jako u služeb. Zde není potřeba procesy vypisovat na základě vstupních požadavků a jsou rovnou vypsány všechny procesy. Pro popis procesu jsou důležité tyto informace: jeho PID, název, velikost paměti (pracovní sada), kterou využívá a také jeho aktuální priorita. U procesů je opět možné

vykonávat dvě činnosti, a to buď proces ukončit, nebo změnit jeho prioritu. Stejně jako u služeb, i zde jsou tyto činnosti, ať už informativní nebo operativní vykonávány určitou sekvencí jednoduchých PowerShell skriptů.

7 Realizace systému

Konzolová verze systému funguje standardním spouštěním skriptů, které již bylo popsáno v několika kapitolách této práce. Pro usnadnění práce uživatele – správce počítačového systému je tak lepší použít grafické uživatelské rozhraní. To bylo zvoleno ve webové formě, která je v současné době k těmto účelům nejvíce využívaná. Webová aplikace je postavena na skriptovacím jazyce PHP a několika dalších rozšířeních, která budou postupně popsána. Implementace systému však začala již před návrhem webového rozhraní a to využitím webových služeb.

7.1 Webové služby

Použití webových služeb vychází z přehledného návodu na implementaci uvedeného na webových stránkách: Webové služby nad IS/STAG v části Technické informace, viz [18]. Zde je uvedena jednoduchá třída v jazyce Java, která webovou službu zpřístupní pro vhodné použití v PowerShellu. Pro potřeby této práce byl vytvořen JAR soubor s rozšířením této vzorové třídy. Rozšíření původně spočívalo v tom, že by aplikace v Javě rozparsovala stažený XML soubor a na výstup by umístila data v čistě textové podobě. Od toho však bylo postupným vývojem upuštěno, zejména proto, že XML dokáže parsovat i PowerShell, což je pro tuto práci efektivnější. Nyní bude krátce popsána implementace webové služby v Javě. Hned na začátku třídy jsou umístěny dvě statické proměnné:

```
public static final String WS_SERVER_URL = "https://stag-
ws.zcu.cz/ws/services";
public static final String WS_SERVICE_PATH =
"/rest/student/getStudentiByPredmet";
```

První z nich obsahuje URL webových služeb STAGu, druhá pak cestu ke konkrétnímu typu a názvu služby. V tomto případě je zcela zřejmé, že se jedná o službu standardu REST, která vrátí studenty nějakého předmětu. Tyto dvě proměnné jsou pak spojeny do výsledného URL a navíc jsou k nim přidány parametry webové služby. Jedná se o zkratku katedry a předmětu, které jsou webové službě předány jako parametry při spuštění této aplikace. Na vytvořené URL je pak pomocí objektu `HttpURLConnection` otevřeno spojení a vykonán HTTP požadavek. V případě,

že jeho výsledkem je hodnota odpovědi 200 (úspěšné provedení požadavku), je pomocí InputStreamu vytvořen získaný XML soubor. V návrhu systému je použita i webová služba pro získání seznamu předmětů pro studovaný obor, kde je zde popsaný princip implementace zcela shodný.

Popis struktury JAR souboru WebService.jar:

- *WebService.java*
  Základní třída zpracuje parametry, které jsou přidány k URL služby a následně proveden HTTP požadavek.
- *XmlParser.java*
  Třída vytvářející XML soubor.
- *User.java*
  Objekt uživatele (studenta) pro ukládání do seznamu pro textový výstup.

7.2 Webové rozhraní

Jako grafické uživatelské rozhraní byla vybrána webová forma z hlediska snadné dostupnosti přes webový prohlížeč. Pro návrh webové aplikace byl vybrán skriptovací jazyk PHP. Přednost oproti ASP.NET nebo JSP dostal zejména kvůli své jednoduchosti a nenáročnosti. Grafický návrh uživatelského rozhraní byl vytvořen pomocí jazyka HTML a kaskádových stylů (CSS). Aplikační logiku pak vytváří právě zmíněný jazyk PHP, který spouští PowerShell skripty tímto způsobem:

```
$cmd = 'powershell.exe -command scripts/script1a.ps1 ...';

$cmd = iconv("UTF-8", "CP852", $cmd);

$pwd = shell_exec($cmd);
```

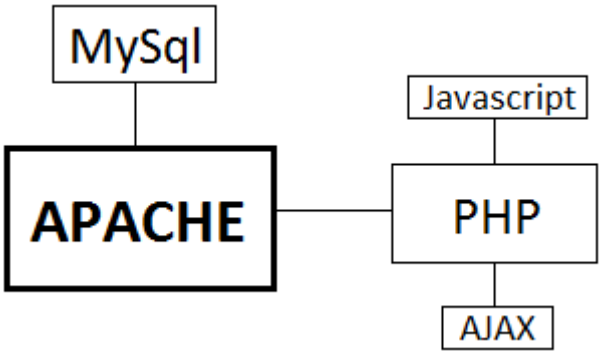
V prvním řádku je uložen příkaz k provedení, ve druhém je provedena konverze znakové sady a ve třetím je příkaz vykonán. Výstupy z PowerShellu je dále nutné parsovat a vhodně poskládat do datových struktur, aby s nimi bylo možné dále pracovat.

Při návrhu webu bylo využito v současné době stále se rozvíjejících lightboxů. Jedná se o zvýraznění konkrétního prvku do popředí stránky a současně ztmavnutí zbytku plochy na stránce. Ač jsou v praxi asi nejčastěji využívány pro zobrazování obrázků, v tomto webovém systému našly uplatnění pro vytvoření různých menu

v rámci modulů. Typickým použitím je zobrazení správy procesů a služeb. Použity jsou také pro zobrazení seznamu předmětů pro přidání studentů do systému MSDN.

Webová aplikace je dále rozšiřitelná. Další modul je možné snadno přidat bez většího zásahu jen přidáním jeho názvu a cesty do datové struktury pole v PHP skriptu definujícím menu aplikace. Automaticky se rozšíří i pracovní okno modulu, které se přizpůsobí počtu položek v menu.

Pro vývoj a testování aplikace byl použit balík XAMPP, který zahrnuje pro tuto práci potřebné součásti: PHP, databázi MySql a server Apache. Databáze je zde použita jen pro přihlášení uživatelů. Forma fungování webové aplikace je naznačena na obrázku 7.1.



Obrázek 7.1: *Běhové prostředí webové aplikace*

Adresářová struktura webové aplikace je následující:

- *kořenový adresář*

  Zde jsou umístěny úvodní stránky: index.php, login.php pro zadání přihlašovacích údajů a utility právě pro přihlášení do systému a následné odhlášení.

- *files*

  Slouží k umístění doprovodných obrázků a kaskádových stylů pro zobrazení webové stránky.

- *js*

  Uložen soubor jquery.js nutný pro použití AJAXU a JSON (kapitola 7.3)

- *other*

  V tomto adresáři jsou umístěny webové moduly pro jednotlivé oblasti činností popsané v kapitole 6.2, které jsou pak vkládány do hlavní stránky systému na základě výběru modulu v menu. Dále je tu ještě podadresář util.

  - *util*

    V podadresáři jsou uloženy PHP skripty pro správu služeb a procesů. Jedná se například o ukončení procesu, zastavení/spuštění služby atd.

- *scripts*

  Zde jsou všechny používané skripty v PowerShellu. Ve stejném adresáři jsou navíc umístěny i JAR soubory používaných webových služeb.

- *utilities*

  V tomto adresáři jsou uloženy PHP skripty pro ukládání standardních výstupů některých modulů do souboru.

7.3 Rozšiřující funkce

Pro lepší design a přehlednost webové aplikace byly použity technologie Javascript, AJAX (*Asynchronous JavaScript and XML*) s využitím JSON (JavaScript Object Notation).

Při zobrazení seznamu předmětů pro výběr studentů pro přidání do systému MSDN je díky Javascriptu možné všechny předměty označit, odznačit nebo invertovat jejich výběr. Následující fragment kódu Javascriptu provede označení všech předmětu v seznamu:

```
function checkedAll () {
    var tmp = document.getElementById('frm1');
    if (checked == false)
    {
        checked = true;
    }
    for (var i = 0; i < tmp.elements.length; i++)
    {
        tmp.elements[i].checked = checked;
    }
}
```

Další použitou technologií je JSON. Ten je z Javascriptu prakticky odvozen a využívá AJAXu pro výměnu dat. AJAX je označení pro technologie vývoje webových aplikací, které mění obsah svých stránek bez nutnosti jejich opětovného načítání. JSON v navržené webové aplikaci našel dobré uplatnění v oblasti správy služeb a procesů. Pro představu využití JSON bude uveden příklad změny stavu služby (zastavení/spuštění) bez nutnosti znovu načítat obsah webové stránky.

```
function changeService(id, service)
    {
        var buttonValue = $("#action_button" + id).val();
        if (buttonValue == "Start") {
            $.getJSON("other/util/startService.php", { service:
            service }, function(data) {
                $("#state"+id).html(data.response);
                $("#action_button" + id).val(data.button);
                if (data.button == "Stop") {
                    $("#restart_button" +
                    id).removeAttr('disabled');
                }
                alert(data.info);
            });
        }
    }
```

Tento fragment kódu vyhodnotí, zda bylo stisknuto tlačítko pro spuštění nějaké služby, na základě jejího ID v tabulce výpisu služeb (nejedná se o ID služby!). Za předpokladu, že tlačítko stisknuto bylo, je odeslán požadavek PHP skriptu ‚startService' s parametrem názvem služby. Na základě provedení tohoto skriptu jsou získána zpět výsledná data, podle nichž se pak změní nastavení služby, s níž je pracováno. Tím je myšleno, že dojde k přepsání hodnoty tlačítka ze ‚Start' na ‚Stop' a též je aktivováno tlačítko ‚Restart', které je při zastavené službě deaktivováno. Správce systému je navíc o výsledku operace se službou informován informačním oknem. Odesílání JSON odpovědi PHP skriptem vypadá následujícím způsobem:

```
echo '{"response": "'.$result.'", "info": "'.$info.'", "button":
"'.$newButtonValue.'"}';
```

JSON odpověď může mít tento tvar:

```
{"response": "běží", "info": "Služba 'Adobe LM Service' byla úspěšně
spuštěna", "button": "Stop"}
```

7.4 PowerShell skripty

Skripty v PowerShellu zahrnují přehledné konstrukce, které jsou navíc v každém důležitém kroku komentovány. Každý z webových modulů obsahuje nejméně dva skripty různé složitosti. Funkce skriptů je různá. Některé musejí poskládat několik navazujících činností, které navíc musí v několika krocích kontrolovat, aby nedocházelo k chybám, jiné skripty jsou zase velmi krátké. Zejména skripty obsluhující správu služeb a procesů patří mezi nejkratší, ale na druhé straně jich je největší množství, protože ovládají mnoho činností a nastavení. Tabulka 7.1 znázorňuje přiřazení skriptů jednotlivým modulům. Názvosloví skriptů bylo zvoleno pro spojení s jednotlivými moduly. Jedinou výjimku tvoří první modul, který byl v prvním návrhu systému rozdělen na dva, a názvosloví skriptů zůstalo zachováno.

Modul	Skript	Stručný popis
Generování skriptů	script1a.ps1	Slouží ke generování skriptu pro vytvoření uživatelských účtů v Active Directory.
	script1b.ps1	Zajišťuje chybový výstup pro předchozí skript.
	script2a.ps1	Obdoba *script1a.ps1* pro vytvoření účtů v SŘBD Oracle.
Správa Active Directory	script3a.ps1	Vytvoření uživatelských kont v Active Directory na základě seznamu studentů konkrétního předmětu získaného z IS/STAG.
	script3aa.ps1	Obdoba předchozího s rozdílem, že je činnost prováděna bez využití rozšiřujícího balíku Quest, jehož funkce nefungovaly při spouštění z PHP.
	script3b.ps1	Získání seznamu uživatelů z Active Directory.
	script3bb.ps1	Obdoba předchozího s rozdílem, že je činnost prováděna bez využití rozšiřujícího balíku Quest, jehož funkce nefungovaly při spouštění z PHP.
	script3c.ps1	Získání seznamu předmětů studovaných na KIV.
	script3d.ps1	Získání seznamu studentů, kteří aktuálně studují předměty KIV.
MSDN	script4a.ps1	Obdoba *script3c.ps1* s tím, že tentokrát se předměty získávají dle studijního oboru.
	script4b.ps1	Získání seznamu studentů vybraných předmětů.
WMI objekty	script5a.ps1	Informace o verzi BIOSu.
	script5b.ps1	Informace o údajích týkajících se pevných disků.
Služby a procesy	script6a.ps1	Získání všech služeb.
	script6b.ps1	Získání aktuálně běžících služeb.
	script6c.ps1	Získání aktuálně zastavených služeb.
	script6d.ps1	Získání automaticky spouštěných služeb.

Tabulka 7.1: *Přehled skriptů přiřazených k modulům*

Pozn.: Poslední modul obsahuje i další krátké skripty, jejichž význam dostatečně popisuje název (např. getProcess.ps1 aj.).

8 Testovací scénáře

Každý z modulů webové aplikace byl otestován ve virtuálním prostředí Microsoft Virtual PC (ver. 6.1.7600) na operačním systému Microsoft Windows Server 2008. Průběh a výsledky jsou shrnuty pro každý modul zvlášť.

Generování skriptů pro vytváření uživatelských kont

Skripty pro vytváření uživatelských kont se generují podle potřebné syntaxe, což je zařízeno skriptem v PowerShellu. Výstup generování je možné uložit do souboru ve tvaru: ‚zkratkaKatedry_zkratkaPredmětu.txt'. Studenti s neexistujícím loginem jsou umístěni do chybového logu. Pokud jsou zkratky katedry a předmětu zadány pro neexistující předmět, nedojde k výpisu generování a příčina je zaznamenána v chybovém logu. Modul fungoval dle očekávání.

Správa Active Directory

Zde už je možností pro testování více. Do prázdné organizační jednotky KIV-Students byla vytvořena uživatelská konta pro studenty nejdříve jednoho a poté třech předmětů. Výsledky testu zahrnuje následující tabulka 8.1:

Předmět	Počet studentů v IS/STAG	Počet vytvořených uživ. účtů v AD
	I. test	
KIV/UPS	133	133
	II. test	
KIV/UPS	133	133
KIV/ZOS	97	145
KIV/UPA	101	159

Tabulka 8.1: *Přehled testu vytváření uživatelských kont*

Z výsledku testů je patrné, že se studenti těchto třech předmětů bakalářského studia prolínají, tudíž celkový počet vytvořených uživatelských kont neodpovídá součtu studentů ve vybraných předmětech. Z konečných 159 vytvořených uživatelských účtů je více jak 100 uživatelů členy více skupin, většinou i všech třech předmětů.

Příklad uživatele:

becvarm@ZCU.CZ – je členem uživatelských skupin: UPS-10, ZOS-10, UPA-10.

V druhé části tohoto modulu, který zajišťuje kontrolu studia studentů, byla pro testování využita omezená skupina studentů. V Active Directory došlo k vytvoření uživatelských účtů pro předměty: KIV/AOS, KIV/NMS a KIV/SPOS. Seznam ze stagu byl pak vytvořen pouze pro poslední dva předměty. Pro potvrzení správnosti funkčnosti by měl být vrácen seznam těch studentů, kteří studují předmět KIV/AOS, ale současně nestudují ani jeden z předmětu KIV/NMS a KIV/SPOS.

Výsledek testu:

havlenor@ZCU.CZ

Jediná studentka studuje předmět KIV/AOS a nestuduje předměty KIV/NMS a KIV/SPOS. Provedené testy proběhly úspěšně.

Předměty KIV studuje velké množství studentů a stažení celého jejich seznamu zabere nějaký čas. Při stahování takto velkého seznamu prostředí PHP vyhodí chybu o překročení doby provedení. Tento problém je možné vyřešit zvýšením hodnoty `max_execution_time` v konfiguračním souboru PHP: php.ini. V průběhu testu byla tato konstanta nastavena na hodnotu 500 sekund a test proběhl úspěšně.

MSDN

V tomto modulu bylo podstatné otestovat dvě činnosti. První z nich je založena na vytvoření seznamu studentů na základě vybraných předmětů a principem zcela odpovídá prvnímu testu pro Správu Active Directory. Jediný rozdíl je zde v tom, že duplicity studentů vyřazuje přímo skript v PowerShellu. Výsledek testu je pro stejně vybrané předměty, jako pro Správu Active Directory, shodný s výsledky uvedenými v tabulce 8.1. Druhá činnost, která kontroluje, zda studenti v systému MSDN stále studují, byla ověřena tak, že k vygenerovanému seznamu studentů pro vybrané předměty byl přidán seznam jiných studentů. Správnost funkčnosti byla dokázána tím, že webový systém nové přidaný seznam oddělil a označil tyto studenty jako nestudující zvolené předměty.

WMI objekty

Protože tento modul má pouze informační význam, jeho test proběhl spuštěním webového systému na více počítačích. Funkčnost byla ověřena změnami zaznamenávaných hodnot týkajících se verze BIOSu a disků a porovnána se skutečnými hodnotami.

Služby a procesy

Poslední modul, zaměřený na správu služeb a procesů, byl testován provedením určité akce a jejím následným ověřením. Pro příklad byly zobrazeny spuštěné služby, jedna z nich byla zastavena, následně byly zobrazeny pouze zastavené služby a námi zastavená služba se mezi nimi objevila. Ověření této funkčnosti dokazuje, že služby jsou získávány v současném stavu stiskem tlačítka ‚Načíst služby'. Stejný princip byl využit i u procesů. U priority procesů byla její změna ověřena znovunačtením seznamu procesů. Ukončování procesů bylo ověřeno spuštěním nového procesu, jehož ukončení nemá nebezpečné následky pro systém nebo práci uživatele (např. TOTALCMD). Stisknutím tlačítka ‚Ukončit' došlo k zavření okna Total Commanderu.

9 Závěr

Tato diplomová práce se zabývá využitím PowerShellu pro usnadnění činností administrátora Windows sítě. V první části práce je představen samotný PowerShell. Je zde seznámení s jeho předchůdci, popsány možnosti jeho použití a analyzovány oblasti, kde může být využit pro správu serverového systému v prostředí katedry KIV.

Pro správu těchto oblastí byla navržena řada PowerShell skriptů, které je možné spouštět z textové konzole, a dále byl pro snazší používání navržen webový systém v PHP, z něhož jsou skripty spouštěny. Skripty v PowerShellu například vytvářejí uživatelská konta v Active Directory, umožňují podporu vytvoření a kontroly účtů v systému správy studentských licencí MSDN nebo nabízejí rozsáhlou správu služeb a procesů. Potřebná data jsou získávána z informačního systému IS/STAG prostřednictvím webových služeb. Data jsou uložena v souborech XML, které jsou PowerShellem zpracovány. Obsahují údaje o studentech nebo studovaných předmětech. Seznam studentů je využit pro vytváření uživatelských účtů, ať už v Active Directory nebo MSDN. Získané předměty slouží pro vytvoření seznamu studentů z většího okruhu, například více předmětů, nebo dokonce celé katedry. Některé skripty jsou použity k získávání informací z WMI. Funkcionalita systému byla otestována testovacími scénáři pro každý modul, výsledky testů jsou shrnuty v kapitole 8.

Navržený a realizovaný systém splňuje účel této práce, je nasaditelný v prostředí KIV a umožňuje snadnou správu zvolených prostředků Windows sítě s využitím PowerShellu. Aplikaci je možné snadno rozšiřovat, případně současné webové moduly modifikovat. Její využití je možné i na jiných katedrách, případně v mimo univerzitním prostředí.

PŘEHLED ZKRATEK

COM (Component Object Model)
model komponent prostředí operačního systému Windows

ADO (ActiveX Data Objects)
soubor COM objektů pro přístup k datovým zdrojům

SNMP (Simple Network Management Protocol)
protokol sloužící pro monitorování a správu sítí

GUI (Graphical User Interface)
grafické uživatelské rozhraní

LDAP (Lightweight Directory Access protokol)
protokol pro ukládání a přístup k datům na adresářovém serveru

DNS (Domain Name System)
služba pro vzájemný převod doménových jmen a IP adres uzlů počítačové sítě

DHCP (Dynamic Host Configuration Protocol)
protokol používaný pro automatickou konfiguraci počítačů v počítačové síti

XML (eXtensible Markup Language)
rozšiřitelný značkovací jazyk

MSDN AA (Microsoft Developer Network Academic Alliance)
software Microsofu poskytovaný akademickým organizacím pro výuku

JAR (Java ARchive)
soubor shrnující několik Java souborů do jediného

AJAX (Asynchronous JavaScript and XML)
technologie využitá u webových aplikací, které umožňují měnit obsah stránek bez opětovného načítání stránky

JSON (JavaScript Object Notation)
odlehčený formát navržený pro výměnu dat

LITERATURA A ZDROJE

[1] MALINA, Patrik, Mgr. Powershell – Podrobný průvodce skriptováním
1. vydání Brno, Computer Press 2007, 345 s.
ISBN 978-80-251-1816-0

[2] How Windows PowerShell works
URL: < http://msdn.microsoft.com/en-us/library/ms714658.aspx >
[cit. 2011-04-12]

[3] Seriál: Windows PowerShell
URL: < http://blogs.technet.com/ >
[cit. 2011-04-16]

[4] MALINA, Patrik, Mgr. Jak vyzrát na Microsoft Windows PowerShell 2.0
1. vydání Brno, Computer Press 2010, 464 s.
ISBN 978-80-251-2732-2

[5] Scripting with Windows PowerShell
URL: < http://technet.microsoft.com/en-us/scriptcenter/dd742419 >
[cit. 2011-05-10]

[6] Running Windows PowerShell Scripts
URL: < http://technet.microsoft.com/cs-cz/library/ee176949%28en-us%29.aspx >
[cit. 2011-05-18]

[7] PowerShell – profil a práce s nápovědou
URL: < http://seven7.blog.zive.cz/2009/12/powershell-%E2%80%93-profil-a-prace-s-napovedou/ >
[cit. 2011-05-20]

[8] VB.NET | Blog – Začínáme s PowerShellem I.
URL: < http://www.vbnet.cz/blog-clanek--319-zaciname_s_powershellem_i_.aspx >
[cit. 2011-05-20]

[9] PowerShell – SAMURAJ-cz.com
URL: < http://www.samuraj-cz.com/serie/powershell/ >
[cit. 2011-05-21]

[10] PowerShell CZ
URL: < http://powershell-cz.blogspot.com/ >
[cit. 2011-05-22]

[11] How to import a Module
URL: < http://msdn.microsoft.com/en
us/library/dd745033%28v=vs.85%29.aspx >
[cit. 2011-05-24]

[12] DUŠEK, Václav – Použití Group Policy pro správu Windows domény
Bakalářská práce ZČU-FAV-KIV
Plzeň, 2009

[13] Windows PowerShell scripting and Administrative Tools
URL: < http://www.quest.com/powershell/ >
[cit. 2011-05-24]

[14] Active Directory Service Interfaces
URL: < http://msdn.microsoft.com/en-us/library/aa772170%28v=vs.85%29.aspx
>
[cit. 2011-05-28]

[15] Služba WMI - přehled
URL: < http://technet.microsoft.com/cs-
cz/library/cc736575%28WS.10%29.aspx >
[cit. 2011-05-30]

[16] Windows PowerShell Tutorial
URL: < http://www.computerperformance.co.uk/powershell/ >
[cit. 2011-06-05]

[17] PowerShell 2.0 commands
URL: < http://ss64.com/ps/ >
[cit. 2011-06-05]

[18] Webové služby nad IS/STAG
URL: < https://stag-demo.zcu.cz/ws/help?page=tech >
[cit. 2011-06-13]

PŘÍLOHY

A – Instalační příručka

Systém lze nasadit na serverový operační systém Microsoft Windows Server 2008, případně Microsoft Windows Server 2008 R2.

K nasazení systému je nutné mít k dispozici tyto komponenty:

- webový server Apache
- MySql databáze
- PowerShell (bez rozšiřujících balíků) – součást Microsoft Windows Server 2008

Ideálním řešením je balík xampp, který obsahuje instalace MySQL databáze, server Apache s PHP a byl využit při vývoji systému.

K nasazení systému stačí umístit adresář se systémem **DIP** (viz příloha C) do adresáře **htdocs** na webovém serveru. V případě využití xampp je tato cesta: **\xampp\htdocs**. Vytvoření uživatele (admin:admin) pro přístup k aplikaci se provede spuštěním SQL skriptu uloženém na přiloženém CD v adresáři: **\DIP\sql**.

Při spuštěném webovém serveru apache pak systém běží na adrese:

http://localhost:port/DIP/

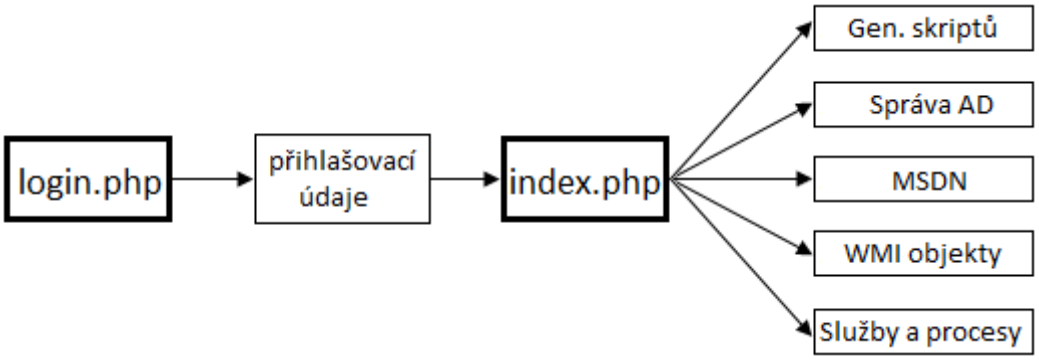
Pozn.: „port“ označuje port, na kterém běží webový server apache

Pro přihlášení do systému je nutné zadat adresu:

http://localhost:port/DIP/login.php

B – Uživatelská příručka

Ovládání webového rozhraní pro správu systému využitím PowerShellu není nikterak složité. Způsob přístupu k aplikaci je znázorněn na obrázku B.1. K zabezpečení systému je použito přihlášení přes uživatelské jméno a heslo. Po přihlášení do systému má administrátor na výběr jednoho z pěti modulů, které zahrnují určitou oblast činností pro správu nebo mají informační charakter.

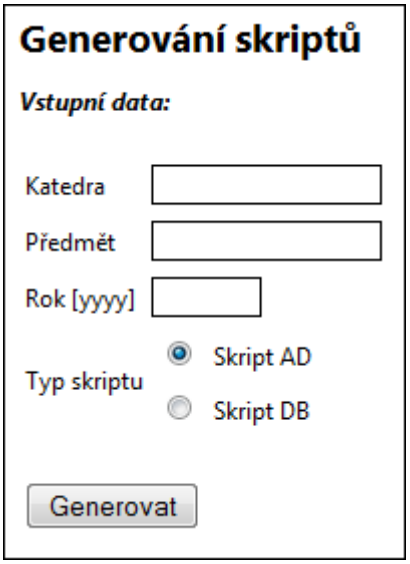


Obrázek B.1: *Přístup k webové aplikaci*

Nyní budou jednotlivé moduly vysvětleny z hlediska vysvětlení jejich použitelnosti.

Generování skriptů pro vytváření uživatelských kont

Zde je ovládání velmi jednoduché. Jak je možné vidět na obrázku B.2 jsou k dispozici celkem tři vstupní pole, kam se zadává zkratka katedry, předmětu a v případě skriptu pro vytváření uživatelských účtů do Active Directory i rok studia. Dále je zde přepínač pro výběr mezi dvěma typy skriptů (AD – vytváření uživatelských kont v Active Directory, DB – vytváření uživatelských kont v databázi Oracle) a tlačítko, které spustí generování. Výsledek generování je zobrazen do výstupního okna v pravé horní části a lze jej dále uložit do souboru stiskem odpovídajícího tlačítko. Pod tímto výstupním oknem je umístěn ještě chybový log, který zaznamenává výskyt chyb při generování.



Obrázek B.2: *Ovládání modulu generování skriptů*

Správa Active Directory

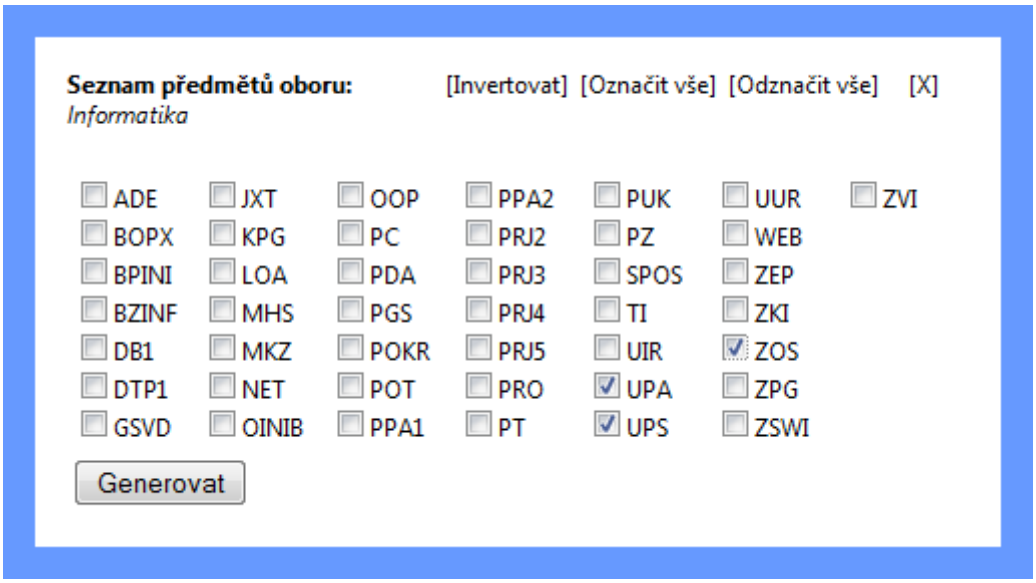
Druhý modul, který spolupracuje přímo s Active Directory má dvě části uživatelského rozhraní.

Levá část slouží k vytváření uživatelských účtů na základě seznamu studentů studujících předmět, jehož zkratka je i se zkratkou katedry zadána do prvních dvou vstupních polí. Zde je určitě vhodné zadat i rok studia, neboť ze zkratky předmětu a právě roku studia je vytvořena uživatelská skupina, jejími členy se stanou právě nově vytvoření uživatelé. Poslední vstupní pole je pro zadání organizační jednotky v Active Directory, do které mají být konta vytvořena. Defaultně je nastavena na KIV-Students, protože tato organizační jednotka je v doméně KIV využívána právě pro přidávání studentských uživatelských účtů. K vytvoření kont dojde stiskem tlačítka ‚Vytvořit uživatelská konta'. Pro případné informace je zde také k dispozici výstupní okno pro log.

V pravé části modulu je systém, umožňující kontrolu, zda studenti, kteří mají svá konta v Active Directory ještě studují. Stisknutím tlačítka ‚Získat seznam studentů ze STAGU' je vytvořen seznam studentů, kteří v současné době studují předměty na KIV a objeví se navíc další tlačítko ‚Porovnat s Active Directory'. Po jeho stisknutí se stáhne druhý seznam, tentokrát s uživateli z organizační jednotky KIV-Students a dojde k jejich porovnání. Studenti přebývající v Active Directory jsou uloženi do souboru ‚ad_compare.txt'.

MSDN

Také tento modul má dvě části. Tou první je přidání uživatelů do systému MSDN. V prvním kroku se zvolí obor studia, z jehož předmětů chceme vybírat. Po načtení se objeví tlačítko ‚Zobrazit předměty'. V okně, které si je možné prohlédnout na obrázku B.3, je možné vybrat ty předměty, jejichž studenty chceme přidat. Výběr předmětů lze různě měnit, invertovat apod. Po stisku tlačítka ‚Generovat' dojde k vypsání studentů ve formě jejich emailových adres do výstupního okna. Současně se ještě objeví nové tlačítko ‚Porovnávací formulář', které po stisknutí zobrazí okno, kam je možné zadat současný stav uživatelských kont v MSDN a porovnat jej s generovaným seznamem. Přebývající studenti jsou uloženy do souboru ‚msdn_compare.txt' Tento mechanizmus slouží k odebrání studentů, kteří již nestudují, z MSDN.



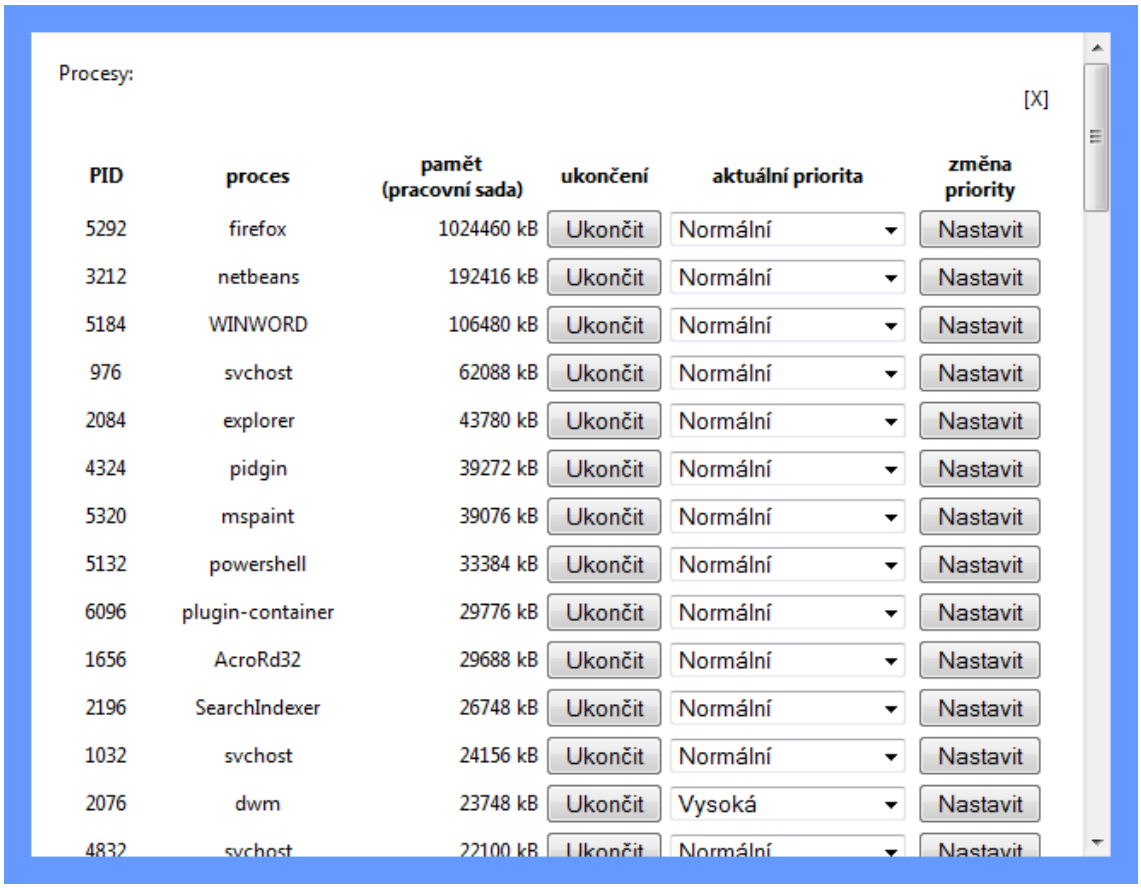
Obrázek B.3: *Okno se seznamem předmětů pro přidání studentských účtů do MSDN*

WMI objekty

Tento modul slouží pro zobrazování systémových informací. Realizované je získat informace o verzi BIOSU a připojených discích. Je snadné doimplementovat zobrazováních dalších informací.

Služby a procesy

Poslední modul má opět dvě části a umožňuje poměrně rozsáhlou správu služeb a procesů. Levá část je využita pro správu služeb. Před jejich zobrazením je možné vybrat, které služby chceme načíst. Na výběr je ze čtyř skupin: všechny služby, běžící, zastavené nebo automaticky spouštěné po startu systému. Po té co jsou služby načteny, již mohou být zobrazeny. V seznamu jsou u každé služby k dispozici potřebné informace o jejím stavu a případných závislostech na dalších službách. Služby je možné stiskem na odpovídající tlačítka spouštět, zastavovat, restartovat nebo měnit mód jejich spouštění. U procesů je to velmi podobné s tím rozdílem, že je není nutné načítat, ale rovnou zobrazit. V seznamu procesů, který si je možné prohlédnout na obrázku B.4, jsou u každého procesu informace ohledně jeho PID a zabírané paměti a také možnost dvou akcí: proces ukončit nebo změnit jeho prioritu.



Obrázek B.4: *Seznam běžících procesů*

Pro ukončení práce ve webovém systému je vhodné se odhlásit odkazem v pravém horním rohu.

C – Obsah přiloženého CD

Přiložené CD obsahuje webovou aplikaci s použitými PowerShell skripty a text diplomové práce; jeho struktura je následující:

- adresář ***DIP*** obsahuje:
  - podadresář **sql** obsahuje skript pro vytvoření uživatele k přihlášení k webové aplikaci v databázi MySql
  - podadresář **scripts** s PowerShell skripty použitými ve webové aplikaci a také webovými službami IS/STAG (soubory *.jar)
  - další adresáře s PHP skripty webové aplikace

- adresář ***WebServices*** obsahuje:
  - podadresář **bin** se spustitelnými JAR soubory webových služeb
  - podadresář **src** se zdrojovými kódy webových služeb
  - soubor **ws.txt** s návodem na použití webových služeb

- soubor ***readme.txt*** obsahuje
  - návod ke správnému nasazení webové aplikace

- soubor ***DiplomovaPrace_Dusek.pdf*** obsahuje text této diplomové práce